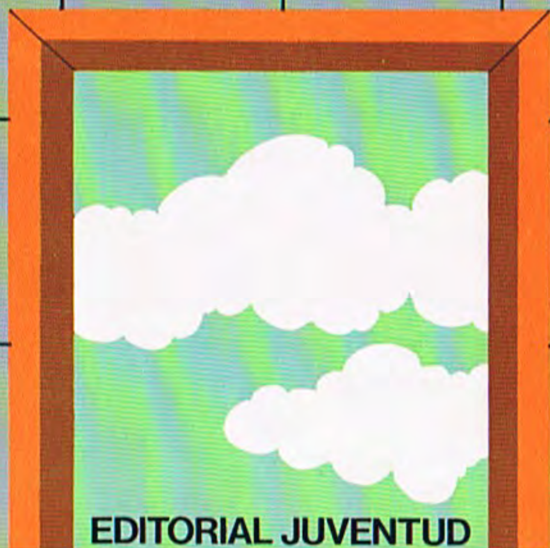
PROGRAMAR EN CASA: EL BASIC

ILYA VIRGATCHIK



EDITORIAL JUVENTUD

Hace algunos años, el título de esta obra hubiera hecho sonreír a los futurólogos y escandalizado a los informáticos... Programar en casa: lujo increíble para unos, imposibilidad teórica para los otros. Hoy, la sonrisa aparece en el rostro de los que programan en su casa y el escándalo es ignorar las posibilidades de la informática individual.

Este pequeño volumen de iniciación y autoaprendizaje del lenguaje Basic está dirigido a los lectores que desean comprender la estructura de un lenguaje informático sencillo y universal.

El objetivo del libro es enseñar a comprender el arte de la programación, y hacerlo de manera que, divirtiéndose, el usuario pueda escribir programas bien concebidos, eficaces y fáciles de modificar.

La obra se completa con un pequeño léxico de los vocablos del Basic. Este léxico habrá de permitirle hallar inmediatamente el significado de una palabra, tanto si pertenece al campo de la informática como al del lenguaje Basic.

Una interesante guía con la que usted podrá descubrir las ventajas del Basic, sus variantes y los Basic especiales.

PROGRAMAR EN CASA: EL BASIC

ILYA VIRGATCHIK

PROGRAMAR EN CASA: EL BASIC

TRADUCCIÓN DE
JAIME GAVALDÁ



EDITORIAL JUVENTUD, S.A.
PROVENZA, 101 - BARCELONA

Título original: PROGRAMMER CHEZ SOI: LE BASIC

Primera edición, 1986
Depósito Legal, B.-23494/86
ISBN 84-261-2180-2
Núm. de edición de E.J.: 7.630
Impreso en España - Printed in Spain

ISCO Impre. S. Coop. C. L.
Pare Font, 125 - 08223-Terrassa

Reconocimiento

Escribir un programa no es nada. Ejecutarlo, todavía menos. El error insospechado acecha al programador a cada pulsación; esto es peligroso.

Nuestro agradecimiento a Christine Morel, profesora, y a Régine de Neve, cuya perfecta relectura nos ha evitado muchos contratiempos.

Introducción

Hace algunos años, el título de esta obra hubiera hecho sonreír a los futurólogos y escandalizado a los informáticos... Programar en casa: lujo increíble para los unos, imposibilidad teórica para los otros. Hoy, la sonrisa aparece en el rostro de los que programan en su casa, en su cocina o en su dormitorio, y el escándalo es ignorar las posibilidades de la informática individual.

Gracias a la «apuesta imposible» de algunos entusiastas del soldador y al dinamismo de los industriales norteamericanos, es ya posible adquirir por poco dinero un microordenador tan potente como los monstruos de los años sesenta. En este libro describiremos brevemente algunos microordenadores de precio reducido. Esto le permitirá, si desea iniciarse en el teclado, hacer una compra que no le decepcionará. La lista de los microordenadores presentados no será, naturalmente, exhaustiva —no es éste nuestro objetivo—, sino que incluirá los «clásicos», cuya calidad ya no hay que demostrar y cuyo precio, competitivo, es consecuencia de una enorme difusión.

Presentación

Este pequeño volumen, iniciación y autoaprendizaje del lenguaje Basic, está dirigido a los lectores que desean comprender la estructura de un lenguaje informático sencillo y universal. Además de la simple iniciación a este lenguaje, esta obra es, pues, una de las claves para comprender el mundo del mañana y las grandes leyes de la programación.

A lo largo de las páginas que siguen, nuestro objetivo será enseñarle a comprender el arte de la programación; y hacerlo de manera que, divirtiéndose, pueda escribir programas bien concebidos, eficaces y fáciles de modificar.

Es aconsejable que utilice este volumen ante la consola de su microordenador, si bien no es indispensable. Nuestra constante preocupación ha sido hacerle comprender las grandes etapas de la elaboración de un programa siguiendo, sencillamente, los ejemplos propuestos. Para progresar y crear usted mismo sus programas, le será necesario un microordenador, aunque durante las primeras etapas de este viaje puede bastarle una hoja de papel y un lápiz.

Sin embargo, para aprender informática —como para aprender a tocar un instrumento musical— es indispensable poder practicar con un instrumento...

Avanzando a través de estas páginas comprobará que, desafortunadamente, no existe un lenguaje Basic único, sino varios «dialectos» Basic que poseen unas características propias. Cada fabricante ha querido dotar al «Basic estándar» de una o varias funciones complementarias. No obstante, la mayoría de las pala-

bras utilizadas —y, en realidad, todas las reglas de programación— son idénticas, cualquiera que sea el Basic empleado. Así pues, el lector que posea un microordenador no tendrá grandes dificultades en adaptar las características del Basic propio de su aparato al Basic descrito en este libro. La lista de las palabras clave que hallará al final de la obra le permitirá resolver los «casos difíciles» sin problemas.

Como el objetivo que perseguimos es ofrecer un contenido lo más didáctico posible, hemos creído útil incluir en este volumen algunos programas prácticos. Nuestra elección ha recaído en programas más específicamente escolares, tales como la conversión grado Celsius/grado Fahrenheit, etc. Este enfoque concreto de los programas nos parece necesario para comprender mejor la utilización de la informática.

Finalmente, nos ha parecido necesario completar la obra con un pequeño léxico de los vocablos del Basic. Este léxico habrá de permitirle hallar inmediatamente el significado de una palabra, tanto si pertenece al campo de la informática en general, como al del lenguaje Basic.

DESCRIPCIÓN DE UN PEQUEÑO SISTEMA INFORMÁTICO

Hardware y software

La máquina en cuyo teclado usted compondrá sus programas en lenguaje Basic y que los ejecutará para usted, es un microordenador. Si su aspecto exterior le es ya más o menos familiar —por haber visto, al menos, su aspecto en las tiendas (por más que actualmente es necesario, a veces, ser muy sabio para distinguir un ordenador de una consola de juegos o de un aparato de televisión con teclado de interrogación a distancia)—, sin duda su «mecánica» interna todavía le resulta un poco misteriosa. A ella vamos a dedicar el presente capítulo.

Ante todo, es indispensable entender bien que su ordenador es una máquina absolutamente estúpida e incapaz de realizar nada si no se le ordena. Estúpida y sin corazón, posee, sin embargo, muchas otras cualidades, que son, principalmente, su paciencia, su infatigabilidad y su rapidez. En efecto, un ordenador es capaz de repetir millares de veces la misma operación sin cansarse, sin presentar signos de irritación y —sobre todo— a una rapidez vertiginosa, varios millones de veces superior a la del hombre mejor entrenado.

• La ejecución de las operaciones en un orden bien determinado recibe el nombre de **programa.** Se distinguen tres tipos de programas:

— Los programas necesarios para el funcionamiento del ordenador propiamente dicho.
— Los programas de ayuda.
— Los programas de aplicación.

Muy pronto hablaremos de estos programas, no sin antes decir que se los opone (son la inteligencia de la máquina) a la máquina misma. Para designar estos dos aspectos complementarios, se hablará de **hardware** y de **software,** que es tanto como decir la máquina y el programa.

Hardware y software son absolutamente complementarios, de manera que el más magnífico de los hardwares de nada sirve sin el software, y, sin software, el mejor ordenador sólo vale la «quincallería» (= hardware) que lo compone.

Por tanto, un ordenador consta de dos partes bien definidas:

- **El hardware,** que comprende no sólo el ordenador propiamente dicho, sino también todos los aparatos que, a veces, se le conectan (son los llamados periféricos del ordenador). El hardware representa, pues, la parte física, tangible, del ordenador.

- **El software** o programa es la parte intelectual, no tangible, del ordenador. El software puede incorporarse a la máquina, ser leído por el ordenador en soportes magnéticos externos (casetes o diskettes) e, incluso, ser creado por el usuario en el teclado de su ordenador.

La parte más extensa de este libro se dedica a un software muy específico —el lenguaje Basic—, que permite al usuario crear sus propios programas.

El software

Es la parte esencial del ordenador. Como ya hemos visto, hay tres tipos principales de software:

1. El software de utilización.
2. El software de aplicación.
3. Los lenguajes.

El software de utilización

Incluso cuando no contiene ningún programa en memoria, el ordenador ha de ser capaz de realizar ciertas tareas, como, por ejemplo, presentar en pantalla la indicación de que está listo (READY, OK, etc.) o cargar en memoria el programa contenido en una casete o un diskette. Si se le conecta una impresora, también ha de ser capaz de enviarle su texto, etc. El ordenador,

desde el momento en que se le aplica tensión, es ya capaz de ejecutar cierto número de tareas que son indispensables para su buen funcionamiento y facilitan su utilización. El programa que ejecutará estas tareas lleva el nombre de programa monitor o sistema de explotación de diskettes (SED).

- **El programa monitor**

Este programa, llamado algunas veces simplemente monitor (en cuyo caso no hay que confundirlo con el monitor, nombre también empleado para designar la pantalla vídeo), es un programa relativamente sencillo que gestiona las entradas/salidas de los pequeños ordenadores, cuya memoria externa es una casete. El programa monitor suele estar contenido en una memoria indeleble del ordenador (memoria llamada ROM, ver más adelante) y es funcional desde el momento en que la máquina se conecta a la red. Por tanto, el usuario no ha de preocuparse por él.

- **El SED**

El sistema de explotación de diskettes, más conocido por sus siglas en inglés **DOS** (Disk Operating System), es un programa monitor mucho más potente. Gestiona ordenadores mayores provistos de diskettes. Este programa está contenido en el diskette propiamente dicho y es el que hace funcionar el ordenador y todos sus periféricos. Sin entrar en detalles, digamos que al principio de la informática cada marca de ordenador tenía su propio DOS. Al cabo de un año o dos, apareció ya una estandarización. Uno de los estándares se denomina CP/M.

- **Los programas de ayuda**

Según la importancia del ordenador, el comprador recibe también algunos programas de ayuda que sirven para leer el lenguaje máquina, recopiar programas, gestionar mejor la pantalla, etc.

El software de aplicación

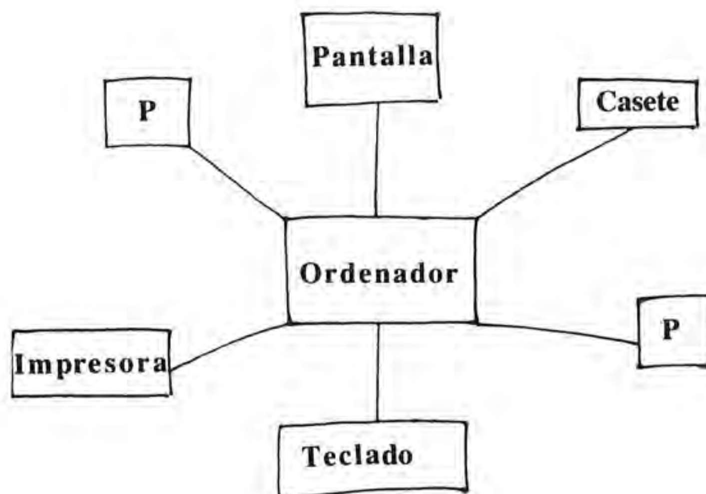
Son los diversos programas específicos que sirven para ejecutar tareas complejas. Son programas «listos para su uso», que se compran en el comercio. Estos programas se escriben sobre soportes magnéticos (diskettes o casetes). Actualmente, pueden hallarse numerosísimos programas de aplicación que abarcan todos los sectores de actividad (programas para las PYME, programas buróticos, etc.) o juegos (ajedrez, bridge...). Algunas veces, estos programas están escritos en lenguaje Basic.

Los lenguajes

Este volumen está dedicado a uno de ellos, el BASIC, pero existen muchísimos otros lenguajes (COBOL, PASCAL, FORTRAN, etc.). Les dedicaremos algunas líneas al estudiar el lenguaje BASIC.

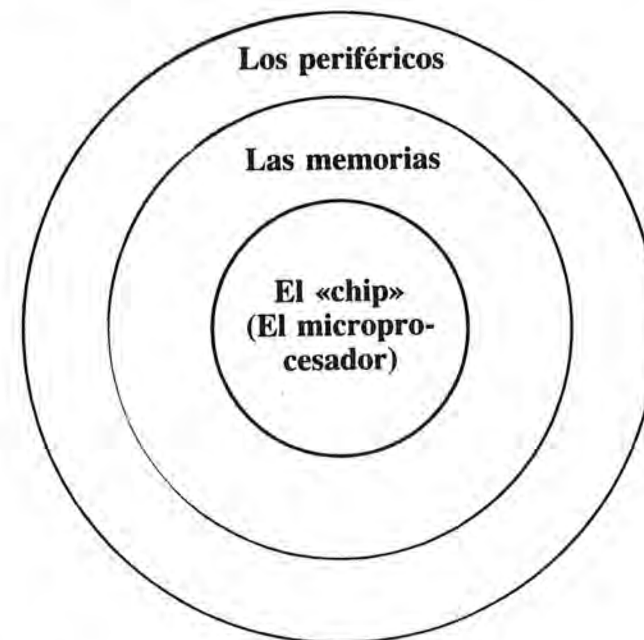
El hardware

El hardware de un ordenador puede ocupar toda una habitación o caber, completo, en el hueco de la mano. Sin embargo, a pesar de esta desproporción, puede decirse a grandes rasgos que se trata de un aparellaje concebido según el mismo plan y compuesto, fundamentalmente, de las mismas piezas. Un sistema informático comprende el ordenador propiamente dicho y una serie de aparatos más o menos sofisticados que gravitan alrededor de aquél (= los periféricos). Estos aparatos aumentan las posibilidades del ordenador y le permiten comunicar más fácilmente con el mundo exterior.



P: Periférico
Pantalla, teclado, impresora... son periféricos

El ordenador está compuesto de un microprocesador y de memorias.



Organización de un sistema informático

Los chips

El **microprocesador,** o **chip,** en inglés, es la base del microordenador.

Este chip, elemento fundamental del microordenador, es también la más pequeña de sus piezas. Se trata, en realidad, de una pequeña pastilla de unos milímetros de lado, pero que contiene millares y millares de transistores, diodos, condensadores, etc.

Por este motivo, se hablará también de **circuito integrado.**

Este chip, insignificante por su peso y por su tamaño, es, sin embargo, más potente que todos los ordenadores de los años sesenta que ocupaban una habitación entera.

Un chip (o circuito integrado) especializado en la gestión o el tratamiento de informaciones se denomina **microprocesador.**

Si a éste se le añaden otros chips especializados en reserva de informaciones (= memorias), se crea un **microordenador.**

El microordenador

Es, ya lo hemos visto, un chip especializado en la gestión y el tratamiento de las informaciones. Está formado por varias partes, todas ellas con una función específica: una, calcula; otra, pone las informaciones en reserva, etc. Se hablará, pues, de unidad aritmética y lógica, de registros, etc.

Para programar en BASIC o utilizar un microordenador es perfectamente posible ignorar esta terminología especializada, terminología que, en cambio, será útil para leer revistas informáticas o consultar los folletos del ordenador.

Entre otras cosas se descubre que hay ordenadores de 8 bits, de 16 bits, etc. En realidad, se trata de ordenadores construidos en base a microprocesadores de 8 ó de 16 bits.

Un **bit** es la unidad de información que corresponde, desde el punto de vista físico, al paso o no de corriente. Un microprocesador de 8 bits es, por tanto, un microprocesador capaz de tratar 8 bits a la vez.

Existen en el mercado numerosos procesadores, todos distintos; pero en microinformática familiar, los ordenadores utilizan casi siempre microprocesadores de 8 bits, como el Z80 de Zilog o el 6502 de Motorola.

Las memorias

Para que pueda ser utilizado, un microordenador ha de ser capaz de poner en memoria el programa y los datos. Véase un ejemplo sencillo: si el programa ordena a la máquina sumar 5 y 7 y luego dividir la suma por dos, la memoria interviene varias veces. La primera, para almacenar las dos cifras; la segunda, para almacenar la suma; la tercera, para almacenar el resultado. En programas más importantes, que es lo habitual, es indispensable disponer de una memoria de gran capacidad.

Hay tres clases de memorias:

— Las memorias auxiliares o externas: son los soportes magnéticos que contienen los programas o los datos (casetes, diskettes).
— Las memorias internas ROM.
— Las memorias internas RAM.

- **Las memorias internas**

La capacidad de estas memorias es más o menos importante según los sistemas. **La unidad de memoria es el kilobyte,** abreviado **K.** Para los microordenadores domésticos destinados a la escritura de pequeños programas en lenguaje Basic, la memoria útil es de ± 16 K.

- **Las memorias internas ROM**

Son memorias que solamente pueden ser leídas. Es imposible escribir en ellas. Por tanto, son interesantes para contener programas utilitarios (programa monitor) y lenguajes (por ejemplo, el Basic). Según la importancia del monitor y del lenguaje, la memoria ROM variará entre 8 y 64 K.

- **Las memorias internas RAM**

Son memorias útiles que aceptan también la escritura. Un pequeño sistema ha de disponer, como mínimo, de 8 K RAM, pero también es posible escribir programas que ocupen menos de 1 K.

Hasta hace poco, las memorias RAM se definían como memorias volátiles, en el sentido de que si se suprimía la corriente, toda la información contenida en RAM se perdía.

Actualmente, muchos pequeños sistemas poseen RAM que guardan la información durante varios días (esto ha sido posible disminuyendo considerablemente el consumo de energía necesario para conservar los datos).

Los periféricos

Ya hemos dicho que un periférico es cualquiera de los aparatos que puede conectarse al ordenador. Según reciba informaciones del ordenador o comunique informaciones al ordenador, se dice que el periférico es de entrada o de salida.

Para que el ordenador pueda comunicar con los periféricos, es indispensable que disponga de un programa especial (*el monitor* o *el DOS*), y, a veces, de un hardware específico. Todo cuanto permite al ordenador comunicar con los periféricos se denomina **interfaz.** Se hablará, pues, de interfaces de hardware y de interfaces de software.

Los principales periféricos de un ordenador son el teclado, la pantalla, la memoria de masa y la impresora. Hay muchos periféricos conectables al ordenador (mesa trazadora, diskettes, etc.), pero no son necesarios para el aprendizaje del Basic.

• **La pantalla**

Muchos ordenadores domésticos utilizan como pantalla el *televisor*. Es interesante saber que, como el aparato de televisión no ha sido concebido para presentar textos, la imagen que da es de calidad mediocre y su utilización prolongada fatiga la vista. Sin embargo, en una primera etapa (iniciación), la pantalla de televisión puede ser recomendable.

Pero cuando empieza a tomarse gusto por la informática (los refractarios son raros) se desea disponer de un *monitor vídeo*. Construido especialmente para dar una imagen estable y contrastada en los pequeños caracteres, permite trabajar varias horas ante el teclado sin excesiva fatiga para los ojos.

Algunos pequeños ordenadores de bolsillo poseen una *pantalla incorporada* que presenta de una a cuatro líneas (20 a 80 caracteres), con posibilidad de pase del texto completo. Estas pantallas de cristales líquidos son muy recomendables para cálculos, pero muy poco para programas largos en lenguaje Basic.

Finalmente, *muchos ordenadores domésticos son capaces de generar color* (de 8 a 256 tonos distintos) en la pantalla de televisión o en el monitor en color. Las instrucciones utilizadas para generar estos colores dependen siempre del fabricante y nada tienen que ver con el lenguaje Basic tradicional, por lo que no será tema a tratar en este libro.

• **Las memorias externas**

Se denominan también **memorias auxiliares** o **memorias de masa.** En microinformática doméstica se utilizará fundamentalmente la casete y, algunas veces, los diskettes.

Estas memorias permiten almacenar informaciones o programas que, a voluntad, pueden pedirse. Cuando, después de varias horas de trabajo, usted haya realizado su primer programa Basic, sin duda deseará conservarlo. Desgraciadamente (salvo excepciones), desde el momento en que apague el ordenador, el programa en memoria desaparece...: por tanto, es indispensable poderlo memorizar en un soporte externo.

El soporte menos caro es la *casete audio*, que todo el mundo posee. Sus principales desventajas estriban en su falta de fiabilidad (los errores de lectura son frecuentes) y en su lentitud (la lectura de un programa requiere, a veces, varios minutos). Además, sólo permite el acceso secuencial, es decir, secuencia tras secuencia.

Si realmente se siente atraído por la informática, muy pronto deseará disponer de una memoria de masa de gran capacidad, lectura rápida y con poquísimos errores de acceso (lectura/escritura): *el diskette*. Desgraciadamente, el empleo del diskette requiere a menudo una inversión mucho más importante:

— compra de un programa de explotación de diskettes;
— compra del lector de diskettes;
— compra (frecuentemente) de un interfaz especial;
— compra de nuevos programas.

Además, el Basic utilizable con diskettes (Disk Basic) necesita también comandos, para los cuales cada constructor ha empleado su propio lenguaje. Al ser este libro una introducción al lenguaje Basic, no tratará el problema —complejo— del Disk Basic.

• **La impresora**

Aunque no es indispensable al principio, muy pronto se verá su utilidad para obtener el listado del programa (cuando sobrepasa algunas líneas, es más fácil leerlo y corregirlo sobre el papel que en la pantalla) o para visualizar los resultados de los programas.

Existen actualmente *impresoras térmicas* (la impresión se hace sobre papel especial, sensible al calor) y *matriciales* (los caracteres, formados por pequeños puntos, se inscriben en una superficie que se denomina matriz) a un precio relativamente moderado. Digamos que, si bien las impresoras térmicas son baratas, el papel es relativamente caro, por lo que una utilización intensiva demostrará muy pronto ser más costosa que la compra de una impresora matricial, que emplea papel normal.

La mayoría de las pequeñas impresoras matriciales permiten también la reproducción de gráficos o dibujos.

Guía de compra

Una progresión lógica (es decir, no dictada por el impulso subjetivo) sería la siguiente:

1. Compra de un microordenador conectable a un TV.
2. Compra de un monitor.
3. Compra de una impresora (matricial o térmica).
4. Compra de un lector de diskettes.

Especificidad de un microordenador

En la actualidad, son muchos los microordenadores que presentan un volumen no mucho mayor que el de las calculadoras de bolsillo. La apariencia resulta engañosa, y entre ambos existe una diferencia esencial.

La calculadora, incluso la de mejores prestaciones, no es capaz de ejecutar más que los programas que tiene inscritos en su memoria. Por este motivo, hay calculadoras financieras, estadísticas, etc. En cambio, el ordenador es capaz de ejecutar cualquier programa, siempre que se le haya entrado en memoria. Por tanto, los ordenadores de bolsillo no sólo disponen de una importante memoria, sino también de un lenguaje de programación. Este lenguaje será muy frecuentemente el lenguaje Basic, si bien, desde hace poco tiempo, van apareciendo pequeñas máquinas programables en lenguaje Forth (ver diccionario al final de este libro).

Finalmente, desde el punto de vista de la terminología, digamos que en la familia de los microordenadores se distinguen las siguientes categorías:

Microordenadores de bolsillo

Se presentan en forma de una cajita del tamaño de una calculadora de bolsillo. Además de una pequeña pantalla de cristales líquidos, poseen teclas alfanuméricas (cifras y letras) y pueden conectarse a una pequeña impresora de bolsillo. Son aconsejables para un aprendizaje del Basic, pero las dimensiones de la pantalla

hacen que esto sea difícil. Los ordenadores de bolsillo tienen su mejor uso en aplicaciones matemáticas o estadísticas.

Microordenadores familiares

Se presentan en forma de un teclado que se conecta al televisor familiar. Su principal representante es el SINCLAIR ZX 81, del que se han vendido centenares de miles de unidades en todo el mundo. Aunque de presentación menos compacta, destacamos también los modelos que a continuación se indican y cuya difusión en todo el mundo es considerable:

— ZX SPECTRUM
— TRS COLOR (Tandy)
— VIC 20 (Commodore)
— COMMODORE 64
— THOMSON T07
— DRAGON
— etc. (ver más adelante).

Estos modelos son aconsejables para aprender a programar en Basic. Las «palabras Basic» utilizadas para la creación de sonidos y colores son indicadas por cada fabricante.

Microordenadores portátiles

Del tamaño de una cuartilla, poseen generalmente un teclado idéntico al de una máquina de escribir y una pantalla que permite visualizar un centenar de caracteres. Admiten numerosos periféricos (diskettes, impresora, etc.). Concebidos especialmente para su utilización durante los viajes, pueden comunicarse (mediante un **modem,** aparato que modula y demodula las señales de manera que puedan transmitirse por cable telefónico) con ordenadores mayores. Los utilizan principalmente los representantes de comercio y los hombres de negocios, pero son poco recomendables para el aprendizaje del Basic o para el desarrollo de programas.

Son varios los fabricantes que presentan estos modelos: Epson, Panasonic, Tandy, New Brain, etc.

Microordenadores individuales

Son los grandes responsables del «*boom* informático». A medio camino entre el ordenador familiar y el ordenador profesio-

nal, se adaptan perfectamente a todas las situaciones. Las dos *vedettes* son el APPLE II y el TANDY modelo I. Aunque tienen más de cinco años (en informática esto supone varias generaciones), todavía no han quedado obsoletos (a pesar de que la fabricación del modelo I de Tandy se ha suspendido y que el Apple II ha sido muy modificado recientemente).

Este tipo de microordenadores es, a nuestro parecer, ideal para aprender informática y el lenguaje Basic.

Microordenadores transportables

Son microordenadores completos que poseen, generalmente, una pantalla integrada (con presentación de 1.000 a 2.000 caracteres) y dos diskettes. Se trata, en realidad, de ordenadores profesionales con un formato compacto (como el de una máquina de coser, con un peso aproximado de 10 kg.).

Como cabeza de lista figura el ordenador OSBORNE, del que se han vendido varios centenares de miles de unidades.

Son muy convenientes para el aprendizaje del Basic, si bien su potencia hace que se destinen a un uso más «serio».

Microordenadores profesionales

Se trata de microordenadores destinados a una utilización profesional. De más prestaciones que los miniordenadores, suelen costar solamente la mitad que éstos. Muchos microordenadores profesionales se construyen en base a una arquitectura de 16 bits. El mercado del microordenador profesional está en plena expansión y la mayoría de los constructores suele presentar uno o varios modelos: APPLE III, TANDY 16, IBM-PC (personal computer), PC-WANG, etc.

Destinados a una utilización profesional (el precio es también profesional...), pueden usarse para el aprendizaje del Basic, pero no es ésta su principal vocación.

Podrá observar —y era éste, precisamente, el objeto de esta breve clasificación— que el término microordenador abarca numerosos productos tan distintos entre sí como pueden serlo una pulga y un elefante.

Presentación del teclado

El teclado le permitirá comunicarse con el ordenador. En principio, parece responder a la misma concepción que el de una máquina de escribir, pero, en realidad, es algo distinto. Nos parece indispensable decir algo sobre su estructura.

Según la disposición de las teclas, existen dos tipos distintos de teclados: El QWERTY, que es el más universal y generalizado, y el AZERTY, utilizado en Francia.

Disposición del teclado

El teclado de un ordenador, además de desempeñar las funciones del de una máquina de escribir, dispone de más teclas que ésta. Las teclas suelen estar agrupadas por bloques, que son:

1. Las teclas alfanuméricas.
2. Las teclas numéricas.
3. Las teclas simbólicas.
4. Las teclas de función.

Es importante saber que, desafortunadamente, no existe un estándar ni para el número de teclas ni para su disposición. Según el ordenador haya sido concebido para tratamiento de textos, para generar gráficos o color, etc., el número de teclas es variable de simple a triple. Algunas de estas teclas, las más importantes, figuran en (casi) todos los teclados. Son las teclas que vamos a describir en breves palabras.

Las teclas alfanuméricas

Poco hay que decir de ellas. Son idénticas a las que tiene habitualmente una máquina de escribir. A destacar que el retorno del carro (lo que en inglés se denomina «carriage return» o, abreviado, «CR»), que indica el fin de la línea, se controla aquí mediante una tecla especial (con la indicación RETURN o ENTER), como la que poseen algunas máquinas de escribir electrónicas.

Las teclas numéricas

En un teclado de máquina de escribir, algunas teclas de la parte superior sirven, a la vez, para las cifras, letras y signos de puntuación. Esto ocurre también en el teclado de un microordenador, pero como habrá que escribir muchas cifras, la mayoría de los aparatos tienen, además, un pequeño teclado (situado a la derecha) que sirve únicamente para las cifras.

• **En cuanto a la cifra 0, hay que hacer una advertencia importante.**

El ordenador no unifica la cifra «0» y la letra «O». Por tanto, es indispensable utilizar siempre bien la tecla numérica cuando lo que hay que pulsar es la cifra. Son muchos los programas que no funcionan porque, simplemente, por distracción o por costumbre el programador ha empleado la letra en vez de la cifra. A fin de distinguirlas bien, **la cifra está siempre barrada: Ø**. Así, mil se escribirá: **1ØØØ**.

Las teclas simbólicas

Además de los símbolos que presentan habitualmente las máquinas de escribir, en el ordenador se encuentran otros, como el signo «mayor que» (>), el signo «sostenido» (#), etc. Más tarde, cuando abordemos el estudio de las funciones Basic, veremos el oficio que cumplen estos símbolos especiales.

Las teclas de función

Estas teclas son muchas, muy importantes, y características de los teclados de ordenador. Es importante ver qué función desempeña cada una de ellas.

Las describiremos en el orden en que habitualmente se presentan:

ESC (ESCAPE): esta tecla sirve para dejar un programa o una función, o para enviar comandos a los periféricos.

CTRL (CONTROL): esta tecla sólo funciona asociada a otras. Estas asociaciones son variables, según las máquinas.

BREAK: es una tecla de paro. Sirve para detener un programa Basic. El programa continuará si se pulsa otra tecla.

SHIFT: es, simplemente, la tecla que permite el acceso a las mayúsculas.

CLS o **CLR (CLEAR):** es la tecla que deja la pantalla libre de todas las inscripciones que contiene.

RETURN o **ENTER:** esta tecla indica, por una parte, el fin de la línea (como el retorno del carro de una máquina de escribir), y, por otra, la necesidad de ejecutar un comando. Generalmente, es la más amplia del teclado.

TAB (TABULADOR): realiza la misma función que el tabulador de una máquina de escribir.

RESET: es la tecla que reinicializa el sistema a cero; hay que utilizarla con prudencia, ya que hace que se pierda el programa en memoria.

Las flechas

Todo teclado de ordenador que se respete posee, al menos, cuatro flechas:
— hacia arriba;
— hacia abajo;
— hacia la derecha;
— hacia la izquierda.

• Estas flechas servirán para desplazar el **cursor**.

El cursor, del que todavía no hemos hablado, es una pequeña señal luminosa y parpadeante que aparece en la pantalla. Indica en qué lugar aparecerá el carácter pulsado. Cuando se modifica un texto, un dibujo o un programa es necesario poder desplazar fácilmente el cursor: ésta es la misión de las flechas. A veces existe una quinta flecha que vuelve a enviar el cursor al principio de la pantalla, a la izquierda.

Ahora que ya conoce usted las principales teclas de un microordenador, podemos empezar a estudiar realmente el lenguaje Basic.

LA PROGRAMACIÓN Y EL LENGUAJE BASIC

¿Programar? ¿Para hacer qué?

Es difícil responder a esta pregunta de una manera sencilla. Programar es adentrarse en un mundo en el que rigor e invención, lejos de oponerse, se complementan. La realización de un programa, es decir, la escritura, según las reglas de un lenguaje, de una serie de instrucciones destinadas a la consecución de un objetivo, exige del programador numerosas cualidades. En efecto, para escribir un programa es necesario, en primer lugar, comprender bien el problema a resolver; y, en segundo lugar, respetar todas las reglas del lenguaje utilizado.

Aparece aquí la diferencia esencial entre un lenguaje informático y, por ejemplo, la lengua española: el lenguaje informático no admite ninguna ambigüedad, no acepta excepción alguna. Es indispensable respetar rigurosamente no sólo la ortografía de una palabra, sino también la sintaxis general de la frase y los signos de puntuación. Una sola coma mal colocada será causa de que el programa mejor redactado no funcione. Interviene aquí el **rigor**.

Para resolver un problema existen, generalmente, muchas soluciones: algunas son rápidas; otras, «elegantes», sin que falten, en fin, las que se escriben fácilmente. En informática, estas tres cualidades no sólo no se oponen, sino que se complementan: el buen programador escribe programas cortos, rápidos y elegantes. Interviene aquí la **invención**.

Por otra parte, el programa ha de ser eficaz. Por tanto, es esencial contemplar todas las posibilidades, todos los «imprevistos» que puedan surgir en el momento del funcionamiento del programa y hallar una solución. Interviene aquí la **creación**.

Así pues, programar es utilizar de manera lúdica todas las cualidades de rigor, creación e invención del programador. Es sorprendente —y a la vez reconfortable— comprobar que ante un ordenador aun los peores estudiantes se manifiestan como brillantes elementos. ¿Por qué? Sencillamente, porque el microordenador —especialmente si está programado en Basic, lenguaje interactivo— es un excelente pedagogo: detecta todos los errores, no cesa de repetirlos, no grita y siempre está del mismo humor (es decir, ¡de excelente humor!). Los buenos resultados son la recompensa inmediata: el funcionamiento del programa.

Entonces, nos dirá usted, ¿por qué programar en casa?

Pues bien, puede programarse por necesidad (para automatizar una tarea repetitiva), por economía (para rentabilizar las horas), por obligación (para adaptar un programa a las propias necesidades), etc., pero también puede programarse —y es lo que le deseamos— por placer. Programar es un pasatiempo maravilloso, como jugar al ajedrez, al *scrabble* o hacer crucigramas... Con la ventaja que aquí no hay necesidad de contrincante ni de encasillados: todo se juega en su intelecto. El microordenador no es, en sí, más que un simple ejecutante.

¿Por qué programar en Basic?

El neófito, deseoso de iniciarse en la microinformática y que entra en una librería para consultar determinadas obras sobre los lenguajes empleados en informática, queda muy sorprendido cuando descubre la existencia de numerosos lenguajes: FORTRAN, COBOL, PASCAL, ADA, LISP, BASIC, etc. Sin embargo, si lo que desea es iniciarse en la programación, se le recomendará una obra dedicada al lenguaje Basic. ¿Por qué el Basic y no otro lenguaje? Vamos a responder a esta pregunta antes de presentar, en algunas líneas, los otros principales lenguajes empleados.

• Pero antes: **¿qué es un lenguaje informático?** Sencillamente, es un medio de comunicación entre el hombre y el ordenador. Si se desea que éste ejecute ciertas operaciones, es necesario ordenárselas. Algunos comandos son manuales, como, por ejemplo, presionar la tecla de puesta en marcha; en cambio, otros pueden indicarse al ordenador sin que sea necesario manipular un determinado botón o pulsar una tecla. De estos comandos se dice que están inscritos en el programa.

Así pues, un programa no es más que un conjunto de instrucciones de comando. Sin embargo, *para que el ordenador comprenda y ejecute el programa es necesario que pueda «comprender» las*

órdenes de ejecución. Como el ordenador no comprende el español, sino solamente un lenguaje supersencillo de una sucesión de paso e interrupción de corriente (es lo que se llama lenguaje máquina), ha habido necesidad de crear un conjunto de convenciones que corresponden a la ejecución de las órdenes. Para facilidad del usuario, se ha dado a este lenguaje una forma parecida a los símbolos matemáticos o a lenguas usuales (como, por ejemplo, el inglés).

Por tanto, *un lenguaje informático es, sencillamente, un conjunto de convenciones* (a los que se ha dado, por ejemplo, la estructura de una palabra) que se utiliza para pedir al ordenador la ejecución de determinadas operaciones.

Para que estas operaciones tengan sentido es obligatorio que se realicen según una **secuencia determinada:** es lo que se llama **programa.**

Los niveles de lenguaje

Ya hemos visto que un lenguaje informático sólo es un medio artificial para comunicar con el ordenador. En realidad, tal como iremos descubriendo, existen diversos niveles de lenguajes, pero todo lenguaje se caracteriza por un punto de vista estático y otro dinámico.

El punto de vista estático comprende la **semántica** (es decir, el significado de las palabras); el punto de vista dinámico comprende la **sintaxis** (es decir, la disposición gramatical de las palabras y su acción recíproca desde el punto de vista del sentido).

Así, estudiaremos la semántica y la sintaxis del lenguaje Basic.

Sería enojoso describir aquí los distintos niveles de lenguaje; digamos tan sólo que se distinguen tres categorías o niveles:

El lenguaje máquina

Es el único lenguaje accesible directamente al ordenador. Todos los demás lenguajes no son más que medios artificiales para facilitar el trabajo del programador y, a fin de cuentas, quedan traducidos a lenguaje máquina.

El lenguaje máquina se compone de una sucesión variable de 1 y de 0 que corresponden al paso o no de corriente.

El lenguaje máquina es propio del ordenador que se emplea y está ligado a la estructura electrónica interna del ordenador.

Es, pues, todo lo contrario de un lenguaje universal: no se utiliza jamás para aplicaciones generales y, naturalmente, nunca por parte del programador principiante.

Los lenguajes de ensamblaje

Estos lenguajes están compuestos de códigos, códigos que son propios de los microordenadores empleados y, por tanto, al microordenador que se utiliza. Los lenguajes de ensamblaje son perfeccionamientos del lenguaje máquina, pero como su empleo es bastante abstracto y complejo, tampoco son utilizados por el programador principiante.

Los lenguajes simbólicos

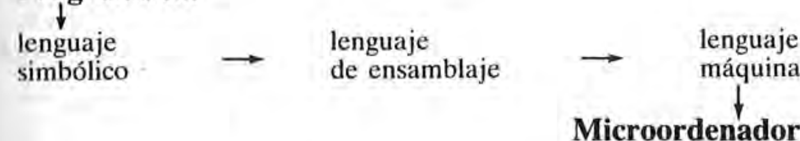
Los lenguajes orientados hacia los problemas, o **lenguajes simbólicos,** se parecen mucho a lo que habitualmente denominamos lenguaje y, por consiguiente, son más cómodos de aprender. Por otra parte, son casi universales, en tanto que pueden utilizarse en todos los ordenadores que los comprenden. Basta con conocer un solo lenguaje para poder programar (a condición, naturalmente, de proceder a algunas adaptaciones menores) en centenares de ordenadores distintos.

Estos lenguajes no son directamente comprensibles por el ordenador y *siempre han de ser traducidos a lenguaje máquina:* se dice de ellos que son **interpretados** o **compilados.** Ya volveremos más adelante sobre el significado exacto de estas dos palabras, pero, a pesar de todo, es interesante saber que esta operación es totalmente transparente para el usuario.

La multiplicidad de los lenguajes simbólicos se explica por las múltiples tareas que puede realizar un ordenador. Algunos lenguajes se adaptan mejor a la resolución de problemas matemáticos, otros son más convenientes para la solución de problemas contables o la creación de música electrónica. Por este motivo, estos lenguajes se llaman **lenguajes orientados** (se dice, por ejemplo, que un lenguaje es orientado gestión) o también **lenguajes evolucionados** (por oposición al lenguaje máquina, muy primario).

Entre estos lenguajes evolucionados, orientados o simbólicos, se halla el lenguaje Basic.

La tabla siguiente ilustra la interacción entre el programador, el lenguaje utilizado y el microordenador:



Origen del Basic

El lenguaje Basic fue creado, hace apenas 20 años (en 1965), en la universidad norteamericana de Dartmouth. Fueron sus autores dos profesores (John Kemeny y Thomas Kurtz) que querían facilitar las cosas a los estudiantes de informática y a todos aquellos que debían tener acceso al ordenador sin ser verdaderos informáticos.

En efecto, estos usuarios tenían que plantear sus problemas a partir de un terminal, pero como sólo tenían acceso a este terminal durante un tiempo limitado (¡la informática era cara!) experimentaban ciertas dificultades para programar sus problemas. Este lenguaje fue creado, precisamente, para ayudarles en la programación.

La idea inicial fue, pues, la creación de un lenguaje muy sencillo, fácil de asimilar, con una semántica parecida a la de la lengua habitualmente empleada (el inglés) y que, además, incluyera fórmulas matemáticas conocidas. A pesar de que su primera vocación fue matemática (lenguaje orientado matemática), el Basic se convirtió, gracias a su sencillez asociada a una gran potencia, en el lenguaje más empleado por los informáticos.

La palabra BASIC es una contracción de Beginner's All-purpose Symbolic Instruction Code. El Basic, como todos los lenguajes informáticos (y como todas las lenguas), ha evolucionado en el transcurso del tiempo y ha adquirido muchas cualidades de las que carecía en sus orígenes.

Al principio, el Basic —que se designa como Basic estándar— era un lenguaje muy pobre (se le llama también Basic mínimo...) y servía principalmente para la utilización de un terminal a tiempo compartido. Para comprender bien la esencia del Basic nos parece necesario aclarar la noción de terminal y la de «tiempo compartido».

Terminal y tiempo compartido

La diferencia esencial entre un terminal y un microordenador estriba en la posibilidad, para este último, de tratar él mismo las informaciones. En cambio, el terminal es incapaz de tratar las informaciones, y para ello ha de recurrir a un ordenador. El Basic, recordémoslo, nació en los años 60, y en esa época se estaba lejos de imaginar que todo el mundo podría poseer un ordenador en su casa. Por aquel entonces, los terminales estaban conectados a un ordenador gigante.

A través de este terminal se podía utilizar la potencia del ordenador gigante de dos maneras:

1. A tiempo compartido.
2. En tratamiento por lotes.

- **El tiempo compartido**

Un ordenador trabaja extraordinariamente aprisa, mucho más aprisa que el intelecto humano o la mano. Por tanto, es posible que varias personas trabajen en un mismo ordenador y cada una de ellas tenga la impresión de que es la única que lo utiliza: el ordenador sólo se interrumpe algunas fracciones de segundo para ocuparse de los otros usuarios. Este cortísimo lapso de tiempo no es percibido por la persona que trabaja ante el ordenador. El tiempo compartido es muy conveniente para todos aquellos trabajos que no exigen ni largas transmisiones ni mucho tiempo de cálculo. En realidad, el tiempo compartido es conveniente para la mayoría de los trabajos de oficina o los propios de los estudiantes.

Precisamente para permitir a los estudiantes emplear de manera más fácil la potencia del ordenador en tiempo compartido se creó el lenguaje Basic. Se deduce de ello que, en su versión inicial, este lenguaje no era recomendable para trabajos largos o para aquellos que exigían una gran potencia de cálculo. Para suprimir esta laguna —que no lo era en el momento en que el Basic se creó— muchos contructores han perfeccionado el lenguaje; esto explica las numerosas versiones actualmente disponibles.

De esta manera, el Basic se ha convertido actualmente no sólo en el lenguaje más utilizado por los no informáticos, sino también en el más empleado por los propios informáticos. Aprender el Basic es, pues, penetrar en el mundo de la informática con paso firme.

Y no es ésta la única razón para estudiar el Basic con preferencia a otros lenguajes: el Basic posee en sí mismo muchas cualidades que descubriremos a lo largo de las páginas de este libro y que es importante destacar desde este mismo momento.

Principales características del Basic

El Basic es: 1) universal; 2) sencillo; 3) potente; 4) interactivo; 5) flexible.

- **Universal**

Significa que el Basic es, a la vez, un lenguaje implantado en todos los ordenadores, y que —excepto en algunos detalles— es idéntico en todas sus formas.

En efecto, cualquiera que sea el sistema que usted decida comprar (desde el más pequeño el más potente), podrá siempre comprar un programa realizado en lenguaje Basic. Además, y a pesar de algunas diferencias (cada fabricante trata de superar a su competidor), la estructura de base a todos los Basic es idéntica. Por consiguiente, después de algunas horas, usted podrá programar en cualquier tipo de ordenador. Más tarde veremos qué es lo que caracteriza las distintas versiones del lenguaje Basic.

- **Sencillo**

El lenguaje Basic —concebido para manipular datos numéricos (cifras) y datos alfanuméricos (cifras y letras)— está orientado, ya lo hemos visto, hacia el usuario. Desde hace algunos años existen también versiones en algunos otros idiomas, aparte el inglés. El francés, por ejemplo, cuenta con su propia versión: el Basicois o Basique.

Las fórmulas matemáticas (que, por otra parte, no es indispensable utilizar) son sencillas y muchos de los problemas corrientes están ya programados.

- **Potente**

A partir del Basic es posible, actualmente, resolver todos los problemas, ya sean matemáticos, contables, de gestión o de enseñanza. El Basic, ya lo veremos, se ha especializado y para ciertos problemas específicos es posible optar por una versión más orientada hacia el problema que hay que tratar.

- **Interactivo**

La interactividad del Basic es, ciertamente, una de sus primeras cualidades, sobre todo para quien se lanza a la programación.

Cuando se dice que un lenguaje es interactivo, significa que entre el ordenador y el usuario se establece un verdadero diálogo.

Así pues, en Basic, cada instrucción es transformada inmediatamente en lenguaje máquina, y cuando el ordenador no comprende la instrucción (ya sea por errónea ortografía de la palabra, ya sea por un error de sintaxis) rechaza el registro y presenta un mensaje de error que indica dónde está dicho error. El programador principiante (y el veterano también) puede, de esta manera, comprobar regularmente su programa.

En otros lenguajes (no interactivos), el programador se ve obligado a escribir, en primer lugar, el programa íntegro antes de poder ensayarlo. Se comprende que esto es muy engorroso, tanto más cuanto que si el programa comprende muchos errores (cosa normal) su corrección puede tardar varias horas o aún varios días. Es algo así como si un pintor o un escultor no pudieran ver su tela o su escultura hasta que estuviera completamente acabada y ¡sólo en el caso de que hubiera salido perfecta!

- **Flexible**

La flexibilidad de un programa es la posibilidad de modificarlo a voluntad y de manera fácil. Por tanto, un solo programa puede servir para diversas aplicaciones.

No es, precisamente, una ventaja pequeña esta posibilidad de poder modificar sin problema alguno un palabra, una línea o un fragmento del programa. Más adelante ya veremos más extensamente lo que esta ventaja supone.

Lo que no se dirá en este libro...

El ordenador en que usted programa es específico.

El Basic de su ordenador es (probablemente) específico. Para hacer trabajar su ordenador, para cargar el lenguaje Basic, para suprimir programas, etc., usted deberá efectuar operaciones de base. Estas operaciones (muy sencillas) varían según los microordenadores. Por tanto, no es cuestión que haya de tratarse aquí. Usted tendrá que consultar el folleto o manual técnico de su microordenador.

Su Basic es un «dialecto» más o menos sofisticado del Basic estándar, incluso es posible que sea capaz de generar sonidos o de dar en la pantalla del televisor color o gráficos. Todas estas posibilidades no existían en el Basic estándar y las órdenes —así como las posibilidades físicas— varían generalmente según los fabricantes.

Para utilizar eficazmente las posibilidades sonoras y gráficas de los microordenadores es necesario, ante todo, conocer bien el Basic estándar. Conociéndolo, no se le planteará ninguna dificultad en el momento de manejar los distintos comandos de su microordenador. Dicho esto —y si pareciere necesario—, la utilización sonora y gráfica de los microordenadores será objeto de un nuevo libro, continuación lógica del presente.

Finalmente, en lo que se refiere a las particularidades de los dialectos del Basic, hemos de hacer tres advertencias:

1. El Basic estándar exigía que cada instrucción apareciera *en una línea separada.* Actualmente esto ya no es así. Sin embargo,

los medios para separar las instrucciones pueden variar según los dialectos (a pesar de que lo más frecuente es el empleo del signo «:»), por lo que hemos adoptado el sistema de emplear siempre una nueva línea para cada instrucción.

2. *Las variables* suelen tratarse de diferentes maneras en los microordenadores. No obstante, todos aceptan dos caracteres y lo tienen en cuenta. En nuestros programas hemos utilizado, pues, variables que no puedan ser confundidas. Así, las variables «ALDEMART» y «ALBERT», al tener como primeras letras «AL» pueden ser confundidas por ciertos microordenadores. Para evitarlo se tomarán como variables, por ejemplo, dos nombres como: «ALDEMART» y «ARTABAN», cuyas dos primeras letras son distintas. O aún mejor: se elegirán variables muy diferentes: «ALDEMART» y «XAVIER». Aquí no hay riesgo alguno de confusión: ni para el ordenador ni para el programador.

3. *Los blancos* (o espacios) no tienen significado en la mayoría de los dialectos; no obstante, intervienen a nivel de la impresión y su empleo hace que la lectura de un programa resulte más fácil.

Estructura general de un programa Basic

Ya hemos visto que un programa no es más que una serie de instrucciones. Pero estas instrucciones han de seguir un orden preciso. Tomemos un ejemplo concreto: una lavadora también ejecuta programas. Según el tipo de ropa a lavar, se empleará un programa con prelavado, con secado o sin él, etc. El programa está concebido con una finalidad: lavar la ropa. Existen, pues, una serie de instrucciones que hay que respetar. El orden de estas instrucciones no es en modo alguno arbitrario. ¿Qué podría pensarse de una máquina que sacara la ropa no lavada? Nada de bueno.

En informática ocurre lo mismo: el programa está compuesto de una serie de instrucciones que han de ejecutarse siguiendo un orden bien determinado. Por este motivo, las línas del programa se numeran. Como hemos adoptado el sistema (ver más arriba) de escribir sólo (como en el Basic estándar) una instrucción por línea, podemos ya enunciar la primera regla:

1.ª regla
Cada instrucción ha de numerarse

Así pues, un programa se presentará siempre de la siguiente manera:

```
10 instrucción
20 instrucción
30 instrucción
etc.
```


Naturalmente, podemos numerar nuestras líneas 1, 2, 3 o también 5, 10, 20, a voluntad. Cualquier número (siempre que sea entero) es válido. Sin embargo, es importante observar que si más tarde queremos añadir instrucciones a nuestro programa y lo hemos numerado 1, 2, 3, será imposible añadir líneas (y, por tanto, modificar el programa).

Se adoptará como regla subsidiaria numerar siempre las líneas de 10 en 10.

2.ª regla
Las instrucciones sucesivas han de ir en orden creciente

Lo que significa que el orden de las instrucciones será el de los números de las líneas. Así, en el programa (ficticio) siguiente:

```
10 ABRIR EL GRIFO
20 TOMAR EL JABÓN
 5 AGITAR LA ROPA
```

Primero se removerá la ropa, y las dos instrucciones numeradas 10 y 20 (abrir el grifo, tomar el jabón) se ejecutarán después.

Esto es debido a que, antes de ejecutar el programa, su ordenador pondrá orden en los números de las líneas de instrucciones. De este modo, la línea 5 (aunque usted la haya escrito en último lugar) será colocada en primer lugar. Esto permite en todo momento añadir instrucciones a un programa y, sobre todo, colocarlas en el lugar debido.

3.ª regla
Cada número debe estar seguido de una instrucción

Por tanto, si deseo que el ordenador presente una frase cualquiera y que después la borre, creo el programa siguiente:

```
10 PRESENTA EN PANTALLA «Yo estoy contento»
20 BORRA PANTALLA
```

Estas tres reglas son la base de la programación Basic: son lógicas y hacen que la lectura de un programa sea cómoda.

La tecla ejecución

Finalmente hay que decir que la tecla ENTER (o EXECUTE o RETURN) ha de utilizarse al final de cada línea.

Esta tecla de ejecución servirá cada vez que sea necesario indicar al ordenador que puede ejecutar una instrucción.

Operación de base

Para comprender estas operaciones de base es necesario decir algunas palabras sobre el funcionamiento general de un microordenador. Recordemos que la unidad central de un microordenador, la que ejecuta todas las operaciones y efectúa todos los controles, se denomina **microprocesador.**

Ya hemos descrito el teclado: es el órgano de entrada mediante el cual se entran las informaciones. Quien entra las informaciones ha de poder leerlas también (para, eventualmente, corregirlas): ésta es la función de la pantalla. Ya hemos visto que se desplaza por la pantalla un cursor parpadeante indicando el lugar donde aparecerá el próximo carácter. Antes de ser presentadas en pantalla, las informaciones (cada vez que usted pulsa una tecla de una información) han de ser tratadas (para, si es necesario, indicarle un error): ésta es la función del microprocesador. Después de su tratamiento, las informaciones se presentan en pantalla: ésta es, pues, un órgano de salida.

Así pues, un pequeño microordenador se presenta de la siguiente manera:

Órgano de entrada → **Microprocesador** → Órgano de salida
(teclado) (pantalla)

• El programa que pulsamos en el teclado es **puesto en memoria** y luego ejecutado. Sin embargo, si cortamos la corriente (o si presionamos la tecla RESET), el programa —y, por tanto, todo nuestro esfuerzo— se pierde. Por consiguiente, es importante poder situar el programa en una memoria auxiliar, que no es volátil. Esta memoria tiene como soporte una casete o diskette magnéticos. El soporte magnético que permite escribir datos y releerlos es, pues, un órgano de entrada-salida.

• Algunas veces se desea también guardar por escrito un programa o los resultados de su ejecución. Para ello es necesario añadir al sistema **una impresora.** La impresora (que tiene la misma función que la pantalla) es, pues, un órgano de salida. Todos los

órganos conectados a un microordenador se denominan **periféricos.** Por consiguiente, podemos expresar la configuración general de un microordenador de la manera siguiente:



• Y los **periféricos de entrada-salida** se dispondrán así:

Órganos de entrada	→ Microprocesador →	Órganos de salida
Teclado		Pantalla
Casetes		Casetes
Diskettes		Diskettes
		Impresora

Comandos Basic de entradas-salidas

Para indicar al ordenador que ha de leer, escribir o imprimir será necesario disponer, en nuestro lenguaje Basic, de las palabras necesarias para los comandos.

Por tanto, ya desde ahora podemos analizar los principales comandos Basic de entradas-salidas.

• Para indicar al ordenador que ha de escribir en la pantalla, el comando es muy sencillo: **PRINT.**

Así, si quiero que en la pantalla se visualice: «Buenos días, me complace estar de nuevo con usted», escribo la primera línea de mi programa de la manera siguiente:

```
10 PRINT «Me complace estar de nuevo con usted»
(a) (b)   (c)
```

(a) corresponde al número de línea (obligatorio);

(b) corresponde al comando: presentar en pantalla (= PRINT);

(c) es el mensaje a presentar: todo mensaje ha de estar entre comillas.

• Para indicar al ordenador que ha de imprimir una línea, el comando es también muy sencillo: **LPRINT.**

De esta manera, el programa siguiente presenta primero la línea en pantalla y luego la imprime:

```
10 PRINT «Me complace estar de nuevo con usted»
20 LPRINT «Me complace estar de nuevo con usted»
```

Más tarde estudiaremos los otros comandos; se hará en el momento oportuno. Sin embargo, para empezar realmente a programar es también necesario describir dos comandos de ejecución.

Comandos de ejecución

Cuando se ha escrito un programa, ha de ejecutarse. Para pedir al ordenador la ejecución de un programa, el comando es muy sencillo: **RUN.**

- Ejemplo: usted ha escrito el programa siguiente:

```
10 PRINT «Me complace estar de nuevo con usted»
20 LPRINT «Me complace estar de nuevo con usted»
```

Luego pedirá al ordenador que lo ejecute pulsando en el teclado **RUN,** y a continuación (no olvidarlo, de lo contrario no pasaría nada) la tecla **RETURN** (o **ENTER**).

- Si su programa es largo y desea obtener de él una lista, basta con pulsar en el teclado **LIST** (luego, **RETURN**) y obtendrá en la pantalla las diversas líneas de su programa. Si, por el contrario, prefiere obtener esta lista en su impresora, habrá que pulsar **LLIST** (luego **RETURN**).

Recapitulación

Ya conocemos ahora algunos comandos Basic. Revisémoslos:
PRINT: presenta un texto en pantalla
LPRINT: imprime un texto en la impresora
LIST: presenta la lista del programa en pantalla
LLIST: imprime la lista del programa en la impresora
RUN: ejecuta el programa.

FAMILIARIZACIÓN CON EL BASIC

Creación de un pequeño programa elemental

En la presentación de este libro se dijo que programar en Basic puede ser una actividad lúdica y proporcionar el mismo placer que jugar al ajedrez o llenar el encasillado de un crucigrama.

En el presente capítulo nos proponemos jugar al Basic creando un pequeño programa capaz de producir *spots* publicitarios como se ven, a veces, en televisión. Este pequeño programa, muy elemental, ha de permitirnos, mientras nos divertimos, revisar algunos comandos del lenguaje Basic. Naturalmente, insistiremos en estos comandos en las próximas lecciones de Basic.

Si dispone usted de un microordenador, podrá crear inmediatamente este pequeño programa y ver los resultados. Si no posee un micro, las reproducciones de pantalla le permitirán seguir este programa paso a paso, como si estuviera delante de su televisor o delante de una pantalla vídeo. Finalmente, a la primera ocasión, ensaye este programa en un comercio de informática. El vendedor le permitirá jugar con los teclados... Un curioso que se divierte tiene madera de futuro cliente.

Primer ensayo

Pulse en el teclado su nombre o cualquier otra cosa, por ejemplo: FELIZ AÑO.

```
FELIZ AÑO
¿QUÉ?
```

Entonces, el ordenador le dirigirá un mensaje a través de la pantalla: WHAT? (o alguna cosa parecida, que significa, a fin de cuentas: ¿qué?) Puede usted comprobar que el Basic es ¡un lenguaje interactivo! El ordenador no comprende qué es lo que desea de él y se lo hace saber. En efecto, FELIZ AÑO no es ningún comando para el ordenador; por tanto, no sabe qué hacer. Recuerde: todo comando Basic ha de estar precedido por un número. Volvamos a probar.

Pero antes suele ser conveniente borrar de la pantalla todo cuanto ha presentado; para ello existe una tecla especial, llamada generalmente **CLEAR** (borrado).

La pantalla aparece de nuevo virgen, lo que nos permitirá efectuar un segundo ensayo.

• No olvide que es necesario **pulsar la tecla de comando de ejecución** ENTER, EXECUTE o RETURN al final de cada línea o después de un comando (como, por ejemplo, RUN).

• **Si el ordenador detecta un error,** presenta un mensaje que indica generalmente la naturaleza del error.

Segundo ensayo

```
10 FELIZ AÑO
SYNTAX ERROR
```

A pesar de la presencia de un número de línea, bien colocado, el Basic no acepta su comando y le indica el defecto: error de sintaxis. En efecto, como ya hemos visto, todo número de línea ha de estar seguido por un comando; ahora bien, FELIZ AÑO no es un comando. El comando para una presentación en pantalla ya hemos dicho que es **PRINT** y el mensaje ha de estar entre comillas. Probemos una vez más.

Tercer ensayo

```
10 PRINT «FELIZ AÑO»
```

No hay mensaje de error; por tanto, la línea debe ser correcta. Así pues, podemos hacer funcionar nuestro pequeño programa. El comando para hacer funcionar un programa es **RUN**.

Pulsemos esta palabra en el teclado:


```
10 PRINT «FELIZ AÑO»
RUN
FELIZ AÑO
```

Este comando tiene como resultado la presentación en la pantalla de la frase FELIZ AÑO. Naturalmente, esta frase queda presentada sin el número de línea y sin comillas.

Si queremos que **en la pantalla aparezca la línea tal como la hemos tecleado,** el comando a utilizar es **LIST.** Este comando nos da la lista del programa. Probemos:

```
10 PRINT «FELIZ AÑO»
RUN
FELIZ AÑO
LIST
10 PRINT «FELIZ AÑO»
```

El comando LIST permite, pues, visualizar el programa tal como fue redactado por el programador.

Creación de un «spot» publicitario

• Antes de empezar a realizar el programa de nuestro *spot* publicitario es interesante saber que el lenguaje Basic posee un comando que no sirve para nada, o mejor dicho, para casi nada. **Este comando indica al ordenador que no ha de tener en cuenta la línea en curso en la ejecución de su programa.** Entonces, dirá usted, ¿para qué sirve este comando? Para poner orden en su programa y hacer los comentarios oportunos. Si más tarde lee usted un antiguo programa o intenta leer un programa escrito por otra persona, estará muy contento de hallar aquí y allá algunos comentarios que indican las intenciones del autor. Este comando, que anuncia las observaciones, se llama **REM.**

Comencemos siempre nuestros programas por esta observación.

• El segundo comando, muy útil, es **el que borra de la pantalla todas las indicaciones.** Como no es posible hacer el borrado con un paño (para eliminar, por ejemplo, todos los ensayos anteriores), un comando realiza esta operación. El nombre de este comando puede variar según los dialectos Basic: **CLS, CLEAR,** etc. Adoptemos el comando CLS, dejando que cada usuario verifique si este comando funciona y, en caso contrario, lo sustituya por el comando de su propio ordenador.

• Conviene empezar todos los programas por estos dos comandos:

REM: para dar un título al programa (cosa que será muy útil en el momento del listado y las eventuales modificaciones), y

CLS: para borrar la pantalla y empezar siempre con una pantalla limpia (además, como el programa comienza indefectiblemente en la parte superior de la pantalla, será posible prever la «compaginación»).

Iniciación del programa

Así, pues, nuestro programa comprende ya dos líneas:

```
10 REM ENSAYO 1
20 CLS
```

Completémoslo con el mensaje a visualizar:

```
10 REM SPOT ENSAYO 1
20 CLS
30 PRINT «FELIZ AÑO»
```

Ejecutemos este programa por medio del comando **RUN** y analicemos lo que pasa:

```
FELIZ AÑO
```

La pantalla presenta solamente el mensaje y nada más. En efecto, la primera línea del programa no desempeña ninguna función, mientras que la segunda borra la pantalla antes de la aparición del mensaje.

Estamos todavía lejos de nuestro *spot* publicitario, pero ya hemos visto los errores a evitar y la manera de presentar un programa.

En cuanto a los errores a evitar, un toque de atención siempre resulta útil (será el último).

Errores frecuentes

1. Olvidar que hay que empezar un programa por un número de línea.

2. Confundir O (la letra) con Ø (cero).

3. Olvidar que el número de línea ha de ir seguido de una instrucción.

4. Olvidar que hay que abrir comillas para todos los mensajes.

5. Olvidar pulsar la tecla RETURN (o ENTER) para terminar todas las líneas y comandos.

Cada vez mejor

Completemos ahora nuestro pequeño programa con una nueva instrucción: **GOTO** (ir a...). Esta instrucción indica al ordenador que ha de ir a otra línea cuyo número está especificado. Así, GOTO 2Ø obliga al ordenador a empezar nuevamente el comando de la línea 2Ø. Naturalmente, sólo puede remitirse al ordenador a una línea que ya existe. Pero dejemos estas sutilezas para más adelante y veamos qué pasa al añadir este comando. Nuestro programa se presenta ahora de esta manera:

```
1Ø REM SPOT ENSAYO 1
2Ø CLS
3Ø PRINT «FELIZ AÑO»
4Ø GOTO 3Ø
```

Ejecutemos este comando (comando **RUN**) y veamos qué pasa:

```
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
FELIZ AÑO
BREAK IN 3Ø
```

El ordenador parece haberse vuelto loco: va presentando en pantalla, sin cesar, FELIZ AÑO. Es necesario pararlo. Pulsemos la tecla **BREAK** (interrupción) y el programa se para en la línea 3Ø (lo cual queda indicado al final de la pantalla con el mensaje BREAK IN 3Ø). Observemos que la tecla BREAK no existe en todos los teclados. Los fabricantes adoptan otras soluciones para detener el programa: CONTROL C, etc...

• En realidad, lo que ha ocurrido es muy sencillo —y, a la vez, muy importante— de comprender.

Después de haber ejecutado la línea 3Ø (que le ordena presentar en pantalla el mensaje FELIZ AÑO), el programa ejecuta la línea 4Ø (que le indica volver a la línea 3Ø y ejecutarla de nuevo, lo que le obliga a ejecutar, a continuación, la línea 4Ø otra vez, etc...).

↑ejecución 3Ø |
| ejecución 4Ø↓

Se trata de un verdadero «bucle infinito». Si nadie detiene el ordenador (o si no se avería), este mensaje se irá presentando hasta el fin del tiempo. Como tenemos prisa, cortamos el bucle deteniendo el programa. Esto se consigue, sencillamente, pulsando la tecla BREAK. El programa se para y en pantalla aparece un mensaje para indicar en qué lugar se ha producido el paro. Ya insistiremos más adelante sobre la tecla BREAK. Por el momento, lo esencial es saber cómo detener el bucle diabólico.

Reemprendamos el programa anterior, pero haciendo una pequeña modificación.

Donde el «spot» toma forma...

Lo molesto del programa anterior era la locura que se había apoderado del ordenador; por otra parte, la sucesión del mismo mensaje no resultaba muy estética. Sería mucho mejor tener el mismo mensaje encendiéndose y apagándose, como un auténtico *spot* publicitario. ¿Habría manera de conseguir este resultado cambiando simplemente una palabra en este programa? ¿Qué piensa usted? Tómese tiempo para reflexionar antes de proseguir la lectura.

• Volvamos al análisis de nuestro programa (sin tener en cuenta la línea REM):

1. La primera línea borra la pantalla.
2. La segunda línea presenta visualmente el mensaje.
3. La tercera línea remite a la segunda.

Así, cuando la pantalla presenta por segunda vez el mensaje, se encuentra en presencia de una pantalla que contiene ya un mensaje; por tanto, se ve obligada a escribirlo a continuación del anterior. Y así sucesivamente hasta que un alma caritativa detiene el programa.

En cambio, si en vez de presentar su mensaje en una pantalla que ya contiene uno lo pudiera presentar en una pantalla virgen, el mensaje aparecería siempre en la misma línea; por tanto, en pantalla sólo se tendría un mensaje único. Para conseguir esto no

hay que resolver ningún grave problema: basta remitir GOTO a la línea 20.

Pues bien, dicho y hecho: modifiquemos nuestro programa.

```
10 REM SPOT ENSAYO 1
20 CLS
30 PRINT «FELIZ AÑO»
40 GOTO 20
```

Ejecutemos este programa (RUN) y veamos qué ocurre.

```
FELIZ AÑO
```

¡Pues no! ¡No es esto! ¿Llegaremos algún día a conseguir lo que deseamos? ¡Claro que sí! Basta reflexionar. El ordenador, ya lo hemos visto, trabaja a una velocidad inimaginable (esto es lo que permite que trabajen varios usuarios simultáneamente y que cada uno de ellos tenga la impresión de que es el único en utilizar el aparato: tiempo compartido), lo que hace que pueda presentar visualmente su mensaje, luego borrar la pantalla y volver a presentar el mensaje sin que usted se dé cuenta.

• Por consiguiente, **para obtener el efecto deseado (flash)** es necesario ocupar el ordenador en otra cosa durante un tiempo más o menos largo.

En el caso presente, le pediremos simplemente que cuente con los dedos (esto, naturalmente, es una imagen) y no haga nada. Para ello, existe en el lenguaje Basic una instrucción muy útil: **FOR... TO.** Esta instrucción es completada con otra: **NEXT.** Esto podría traducirse como: «para... hasta» «repite». Esta doble instrucción puede tomar numerosos valores; por ejemplo:

```
10 FOR X = 1 TO 100
20 NEXT X
```

Lo que se traduciría por: **PARA X = 1 a 100 REPITE X** o, en otras palabras, *cuenta de 1 a 100* (sin hacer ninguna otra cosa).

• Este bucle se llama **bucle de temporización.**

• Se emplearán distintas letras (como X, Y, Z, etc.), ya que estas letras (que se llamarán **variables**) pueden tener un valor. Pero esto es ya mucho más complicado y no merece la pena discutirlo ahora.

• Debemos, pues, **añadir dos líneas a nuestro programa,** ya que hemos decidido poner una sola instrucción en cada línea.

Estas líneas serán N.º línea FOR X = 1 TO 500.
N.º línea NEXT X

Según usted, *¿en qué lugar hemos de colocar estas líneas?* O, en otras palabras: ¿qué números habremos de darles? Reflexione antes de leer lo que viene a continuación.

Respuesta: el único lugar útil se halla entre los dos comandos que nos interesan (es decir, CLS y PRINT), o sea, en las líneas 25 y 26. Si no me cree, nada le impide hacer otros ensayos...

Evidentemente, podemos numerar estas líneas como 23 y 24 o 26 y 27, pero es bueno seguir la regla de renumerar de 5 en 5 (por las mismas razones por las que habíamos decidido la numeración de 10 en 10).

Así, pues, nuestro programa toma la siguiente forma:

```
10 REM SPOT ENSAYO 1
20 CLS
30 PRINT «FELIZ AÑO»
40 GOTO 20
25 FOR X = 1 TO 500
26 NEXT X
```

Hagamos un listado (es decir, pidamos al ordenador que nos presente visualmente todo el contenido del programa) de este programa y veremos que las instrucciones vuelven a ponerse en orden. El listado del programa se obtiene pulsando en el teclado LIST:

```
10 REM SPOT ENSAYO 1
20 CLS
25 FOR X = 1 TO 500
26 NEXT X
30 PRINT «FELIZ AÑO»
40 GOTO 20
```

• **Ejecutemos este programa (RUN),** y veamos qué pasa (aquí es necesario hallarse ante una pantalla para ver lo que ocurre: el mensaje aparece, luego desaparece, como el *flash* de un *spot* publicitario).

```
FELIZ AÑO
```

• **Modificando el valor de los bucles de temporización** (FOR X =

1 TO 500 y NEXT X) se pueden provocar *flashes* distintos (más lentos, más rápidos).

En realidad, sólo se trata de un bucle que está comprendido entre la instrucción FOR TO y la instrucción NEXT (que siempre ha de terminar un bucle):

FOR (palabra clave = variable) **TO** (da el valor)	→	**bucle de temporización**	→	**NEXT** (palabra clave = variable) (termina el bucle)

• Volveremos a tratar extensamente de los bucles más adelante. Sin embargo, no será inútil recordar que **ya hemos estudiado dos clases de bucles**:

1. GOTO
2. FOR... TO NEXT

El primer bucle dejaba al programa en total libertad y no se sabía cuántas veces estaría activo este bucle (en nuestro ejemplo, hasta el fin de los tiempos...).

El segundo bucle obliga al programa a efectuar cierto número de operaciones.

Los bucles desempeñan una función muy importante. Por tanto, volveremos a hablar del tema con más detalle.

Introduciendo otros bucles de temporización se puede hacer variar el tiempo de visualización del mensaje, el del borrado de la pantalla, etc.

Nuestro propósito era poderle mostrar la manera de crear, divirtiéndose, un pequeño programa que exige, sin embargo, la utilización de numerosas instrucciones.

Esta primera familiarización con el lenguaje Basic nos ha demostrado que programar no es muy difícil (incluso si se hace como diversión). Ahora, podemos ya atacar de frente el lenguaje Basic.

OBSERVACIONES GENERALES ACERCA DE LOS PRINCIPALES BASIC

Las variables, las constantes y el modo directo

Para una buena comprensión, es necesario aclarar desde este momento qué significan estas palabras.

• **Una constante** es un valor que no cambia: la cifra 7 es una constante.

• **Una variable** es un valor fluctuante.

En su memoria central, el ordenador reserva determinadas «casillas» para colocar en ellas cifras y letras que no se fijan de una vez por todas (como, por ejemplo, la temperatura del día o las previsiones meteorológicas). Estos valores fluctuantes se denominan variables. Para reconocer las distintas variables que va colocando en las «casillas» de su memoria, el ordenador ha de asignarles un nombre. Lo más corriente es que sea una letra de alfabeto. Es *importante* saber que:

Las variables numéricas (por consiguiente, las cifras) **se identifican por letras** (A, B, C...).

Las variables alfanuméricas (por consiguiente, las cifras y las letras) **se identifican por letras a las cuales se fija el símbolo dólar** (por ejemplo, A$, Z$,...).

• **Un ordenador puede utilizarse en modo directo;** es el modo más

simple y el ordenador se comporta como una gran calculadora: se pulsan una o más teclas en el teclado y la pantalla presenta el resultado. El modo directo se opone al modo programa, objeto de este libro.

Las principales diferencias entre los lenguajes Basic

Las principales diferencias entre los lenguajes Basic están en el tratamiento de las cadenas de caracteres («ANTONIO» es una cadena de caracteres), en la manera de acceder a las memorias externas (casetes o diskettes), así como en el modo de gestionar los colores y los sonidos. Esto se explica fácilmente recordando que el primer Basic no fue concebido para adaptarse a los ordenadores individuales (que, por otra parte, en aquel entonces no existían) ni tampoco para utilizar colores y sonidos. Digamos también que los Basic suministrados con las consolas de juegos se apartan considerablemente del Basic estándar. Esto es comprensible porque estas consolas están fabricadas para explotar al máximo las posibilidades lúdicas creadas por los colores y los sonidos, posibilidades tampoco explotadas por el Basic estándar.

Ya veremos más adelante la manera en que los distintos Basic tratan las cadenas de caracteres y el acceso a las memorias externas.

¡Oh, el bonito cero!

En comunicación electrónica, era tradicional barrar la cifra cero para distinguirla de la letra «O»; desafortunadamente, los primeros informáticos desconocían esta regla y era muy frecuente que barrasen la letra O y no la cifra. Por tanto, esta costumbre desapareció y sus huellas solamente pueden hallarse en la lectura de programas antiguos.

• Recuerde, pues, que **la cifra cero ha de ir siempre barrada con una barra diagonal** (∅).

• Para evitar contratiempos (frecuentes sólo cuando se redacta un programa) acostúmbrese a **evitar la letra O** como si se tratara de la peste; sobre todo, no la emplee nunca en el nombre de cualquier variable ni, preferentemente, en el nombre de un programa.

Las categorías de las palabras

En lenguaje Basic aparecen cuatro categorías de símbolos y de palabras. Tendremos ocasión de describir unos y otras cuando estudiemos el lenguaje, pero es interesante distinguir ya desde ahora estas categorías en función del papel que desempeñan en el lenguaje:

Las cuatro categorías son:

1. Las palabras de comandos.
2. Las palabras de declaraciones.
3. Las palabras de funciones.
4. Los símbolos operadores.

• **Las palabras de comando**

Estas palabras, como muy bien indica su categoría, **ordenan al ordenador efectuar un comando cualquiera.** Este comando puede referirse al programa (por ejemplo, ponerlo en marcha: RUN, o dar su listado: LIST) o también a uno de los periféricos (accesorio) conectados al ordenador (por ejemplo, poner en marcha el lector de casetes).

• **Las palabras de declaración**

Son palabras que se refieren al programa propiamente dicho: **son las distintas instrucciones del programa.** Las declaraciones del lenguaje Basic primitivo eran poco numerosas. Actualmente, cada constructor desea aumentar las posibilidades de su máquina; por tanto, ha de aumentar el número de instrucciones con el consiguiente incremento de las palabras de declaración. Entre estas últimas podemos empezar a mencionar: PRINT (para la visualización en pantalla), GOTO (para crear un bucle), LET (para dar un valor a una variable), etc.

• **Las palabras de funciones**

Gracias al empleo de calculadoras de bolsillo, todos nos hemos habituado a lo que puede efectuar una máquina pequeña, es decir, una serie de operaciones complicadas como, por ejemplo, calcular la raíz cuadrada de un número o el valor de pi. En las calculadoras, basta pulsar una tecla preprogramada. En un ordenador, **es necesario pulsar en el teclado el nombre de la función.** Por ejemplo, para calcular el logaritmo de un valor A, se pulsará en el teclado LOG (A).

• **Los símbolos operadores**

Son los diversos símbolos que **ordenan al ordenador efectuar determinadas operaciones.**

Los principales símbolos operadores (que, por otra parte, ya le son familiares gracias a las matemáticas) son +, −, > (mayor que), < (menor que), etc.

El editor

La palabra editor es una traducción (¿desafortunada?) del vocablo inglés «editor», que en esta lengua indica una actualización, una modificación.

• **Un programa editor es, pues, un programa que permite fácilmente la modificación de otro programa;** en nuestro caso, de un programa Basic.

• Cuando usted escriba sus programas o cuando transcriba los de una revista, se dará cuenta muy pronto que **nada es tan fácil como cometer errores.** Poner dos T en print: PRINTT, o colocar una coma en vez de dos puntos son errores que, en Basic, son imperdonables: el programa no funcionará. Será totalmente indispensable corregir estos errores. Naturalmente, siempre es posible volver a escribir totalmente la línea. Si sólo se escribe un comando por línea (como en Basic tradicional), esto no es muy grave; en cambio, si son varios los comandos escritos en una misma línea (cosa que permiten la mayoría de los «dialectos» del Basic), resultará muy engorroso tener que escribir de nuevo una línea entera para corregir un simple error de puntuación.

• **Gracias al programa editor es posible desplazarse por el programa para modificar, suprimir o insertar una palabra o un símbolo.**
Por tanto, un programa editor es un instrumento que facilita las cosas al usuario. Cada máquina posee su propio programa editor, por lo que no es necesario tratar aquí este tema.

• Digamos, simplemente, que **el programa editor se pide generalmente pulsando en el teclado la palabra** EDIT, seguida del número de la línea a corregir. Este comando pone al programa en modo editor, lo que permite corregir el error con toda facilidad. También es interesante saber que muy frecuentemente el Basic se pone por sí mismo en modo editor cuando constata un error en la programación.
Para aclarar bien los conceptos, tomemos un ejemplo: usted compone en el teclado la línea siguiente: 1Ø PRINTT «SOY FELIZ».
Inmediatamente, el ordenador hace una presentación visual:

SYNTAX ERROR y, en la línea siguiente, aparece la cifra 1Ø. El ordenador se ha puesto a sí mismo en modo editor. Pasa exactamente lo mismo que si usted hubiera pulsado en el teclado EDIT 1Ø. Esta función —que puede hallarse en la mayoría de los Basic— facilita la labor del programador.

• Recordemos una vez más que, en Basic, el ordenador detectará algunos de los errores que usted cometa, pero esto no es una garantía de perfecto funcionamiento de su programa. Puede ocurrir que la sintaxis sea perfecta, pero **es posible que existan errores de lógica que no serán detectados por el ordenador.**
Esta pequeña advertencia sobre el editor nos lleva a decirle también algunas palabras respecto al tratamiento de textos (confundido algunas veces con el editor de textos) y a los mensajes de errores.

El tratamiento de textos

El tratamiento de textos es un programa que tiene por objeto principal permitir al usuario escribir en su teclado un texto sin preocuparse en absoluto ni de su disposición ni de los errores cometidos. Una vez terminada la escritura, podrá tratar el texto inmediatamente, es decir, podrá modificarlo, añadirle o quitarle letras, palabras o frases. También tendrá la posibilidad de desplazar párrafos, crear márgenes automáticos, sustituir algunas palabras automáticamente por otras, etc.

• **El tratamiento de textos es, pues, un programa de ayuda a la escritura** (en este vocablo queda englobada la concepción, la realización y las múltiples impresiones). Por consiguiente, **se trata de un programa que actúa sobre el texto.**

• En cambio, **un editor de textos es una ayuda a la corrección de un programa.** Por consiguiente, se trata de un programa de ayuda a la programación.
Recordemos bien esta diferencia:
— editor de textos: ayuda a la programación.
— tratamiento de textos: ayuda a la escritura.

Los mensajes de error

Ya hemos visto que el Basic es un lenguaje interactivo. Y lo que es más: no sólo indica al usuario que está cometiendo un error, sino que también suele ponerse en modo editor.

• Para ayudar al programador a corregir su error, **es frecuente que le indique la naturaleza de dicho error** mediante un mensaje. Este mensaje de error puede ser un código (de cifras o letras) o, mejor aún, una clara indicación (aunque, desgraciadamente, suele aparecer en inglés).
• Los principales mensajes de error se refieren a los **errores de sintaxis** (por ejemplo, dos comas, una a continuación de otra), **olvidos** (por ejemplo, un GOTO a una línea que no existe), etc.
• Es interesante observar que los mensajes de error no existen en el Basic estándar, por lo que **no hay reglas universalmente aceptadas.** Así pues, conviene consultar el manual que acompaña al ordenador. La actual tendencia de los fabricantes (tendencia que hay que aplaudir sin reservas) es hacer que los mensajes sean lo más numerosos y lo más explícitos posible.
• Después de la lectura de este capítulo, el lector habrá comprobado que nunca se verá totalmente abandonado cuando se halle ante el teclado (incluso si lo ignora casi todo del Basic). En los últimos años **se han desarrollado muchísimas ayudas a la programación.** Pero a pesar de estas ayudas, el programador es, en definitiva, señor absoluto y el programa sólo será, a fin de cuentas, un reflejo de su rigor, de su inteligencia y de su creatividad.

LAS PRINCIPALES INSTRUCCIONES DEL BASIC

Para empezar a programar en lenguaje Basic no es necesario aprender centenares de instrucciones o palabras nuevas. Una veintena de vocablos bien elegidos (que se hallan en todos los «dialectos») son suficientes para la creación de numerosos programas.

Esta lista de las palabras más empleadas en Basic es la que le proponemos aprender y utilizar. Cuando conozca bien el empleo de estas palabras, le será fácil añadir a la lista de base las instrucciones especiales de su microordenador personal y completar así sus programas con instrucciones gráficas, sonoras, de tratamiento de cadenas, etc.

• Las palabras que sirven para designar comandos o instrucciones se denominan **palabras reservadas.** Las palabras fundamentales del Basic son:

Comandos

AUTO	DELETE
CLEAR	EDIT
CLOAD	LIST
CONT	NEW
CSAVE	RUN

Instrucciones

LET	GOSUB RETURN
INPUT	FOR TO NEXT
PRINT	READ DATA RESTORE
GOTO	END
IF THEN ELSE	DIM

• Además de las palabras reservadas de comando o de instrucción, el lenguaje Basic emplea también algunas palabras para designar **funciones,** como, por ejemplo, COS, para calcular el coseno de un valor angular.

• Otra palabra, muy utilizada, es REM, que sirve «sencillamente» para indicar al ordenador que lo que sigue sólo es una **observación.**

• Finalmente, el lenguaje Basic emplea también **operadores, constantes y variables.**

Vistos los principales símbolos operadores, dedicaremos algunas páginas a la descripción de las variables.

Así, pues, el capítulo se dividirá en cinco partes:

1. Las palabras reservadas
2. Los operadores
3. Las constantes y las variables
4. Diagrama de flujo
5. Últimos consejos

Las palabras reservadas del Basic

Las palabras reservadas son aquellas que en el lenguaje Basic tienen un significado cualquiera (comandos, declaraciones, etc.) para el ordenador y en el programa. Se las llama palabras reservadas porque, en principio, **no han de utilizarse jamás en una variable,** so pena de provocar errores de programación.

Describiremos en este capítulo las veinte palabras más importantes. Cada palabra se ilustrará con un ejemplo. Más adelante encontrará ejemplos de aplicación en los programas propuestos en ejercicios. Finalmente, hay otras palabras menos importantes que se explican en el diccionario del Basic que figura al final de esta obra.

Las palabras de comando

Son palabras que ordenan al aparato efectuar un comando cualquiera en el programa; por tanto, no forman parte del programa propiamente dicho, sino que se pulsan directamente en el teclado. Ya hemos visto varias palabras de comando como, por ejemplo, **RUN,** que indica al ordenador que ha de ejecutar el programa.

Hay que advertir que *algunos lenguajes Basic admiten palabras de comando en el programa: estas palabras se convierten entonces en palabras de declaración.* Así, el comando RUN en un programa ordenará al aparato la ejecución, en un momento deter-

minado, de otro programa. Por consiguiente, se pueden crear muchos pequeños programas que se ejecutan los unos a los otros.

AUTO

Este comando, presente en la mayoría de los Basic, *realiza una numeración automática de las líneas*, permitiendo una programación más cómoda. Por tanto, basta pulsar la línea del programa sin preocuparse de su número.

El comando puede presentarse de tres maneras diferentes:
— sin argumentos: AUTO
— con un argumento: AUTO 15
— con dos argumentos: AUTO 15,7

- **Sin argumentos:** provoca una numeración automática de 1Ø en 1Ø. La primera línea presentada es la 1Ø; cuando se termina esta línea (y pulsando la tecla RETURN o ENTER) presenta la línea 2Ø, etc. La numeración se interrumpe pulsando la tecla BREAK o ESCAPE.
- **Con un argumento:** empieza la numeración en la línea correspondiente al argumento. El incremento sigue siendo de 1Ø. Ejemplo: AUTO 15 provocará la presentación visual de las líneas 15, 25, 35, etc.
- **Con dos argumentos:** ordenará la numeración en la línea correspondiente al primer argumento con un paso que viene fijado por el segundo argumento. Ejemplo: AUTO 17,4 provocará la presentación visual de las líneas 17, 21, 25, 29, etc.

CLEAR

Este comando actúa de manera diferente según los ordenadores. En algunos, **provoca el borrado del programa inscrito en la memoria interna** (RAM); en otros **actúa solamente sobre la memoria afectada por las variables.**

Para ver si este comando actúa sobre toda la memoria, introduzca un pequeño programa de sólo dos líneas; pulse LIST (el programa ha de quedar listado en la pantalla) y luego CLEAR. Si no aparece nada más, CLEAR ha borrado el programa de la memoria interna. En la mayoría de los sistemas, esta función no la realiza el comando CLEAR, sino NEW (ver más adelante).

Lo más corriente es que la función de CLEAR sea otra: **reponer las variables a cero.** Más adelante trataremos con detalle de

las variables, pero el pequeño programa que se da a continuación le permitirá apreciar la función de este comando.

1Ø LET A = 7 2Ø PRINT A	Éste es el programa (se pide al ordenador que guarde en memoria que la variable A valdrá 7).
PRINT A Ø	Éste es un comando directo. Se pide al ordenador que presente visualmente el valor de A que posee en memoria. Como el programa todavía no se ha ejecutado, la respuesta es, naturalmente, cero.
RUN 7	El programa es ejecutado y presenta visualmente el valor de A: 7.
PRINT A 7	Es un comando directo. El valor que se visualiza es ahora 7. En su memoria, el ordenador ha anotado que A valdrá 7 hasta que no se le diga lo contrario.
CLEAR	Este comando repone todas las variables a cero (retorno al punto de partida).
PRINT A Ø	

CLOAD (o LOAD)/CSAVE (o SAVE)

Una vez determinado el programa, suele ser útil poder conservarlo en otro lugar que no sea la memoria central (que, recordémoslo, es volátil). El soporte ideal (por su facilidad y buen precio) para el ordenador familiar sigue siendo la casete. Los soportes externos baratos no existían en el momento en que se ideó el primer Basic. Por tanto, las palabras utilizadas para controlar las casetes varían en función de los fabricantes: **LOAD** y **C(asete)LOAD** son las más frecuentes. **El comando CSAVE permite cargar un programa contenido en memoria central en una casete.** Naturalmente, para que el ordenador pueda distinguir (y más tarde releer por medio del comando CLOAD) este programa entre los millares que usted escribirá, es indispensable que le dé un nombre (es lo que se llama, también, una etiqueta, o un label). Este nombre ha de estar compuesto por un máximo de 8 letras y ha de ir entre comillas.

- **Ejemplo:** CSAVE «JUVENTUD»

El programa contenido en memoria central será **«conservado»** (grabado) en casete bajo el nombre de «JUVENTUD». Si, más tarde, se desea **«cargar»** (leer) el programa en memoria central,

bastará insertar la casete en el lector y pulsar CLOAD «JUVENTUD». Si son varios los programas que figuran en la casete, el ordenador podrá reconocerlos gracias al label, la etiqueta, y sólo conservará en memoria el programa que lleva el nombre de «JUVENTUD».

CONT

Si la ejecución de un programa se ha detenido y se desea continuarla, basta pulsar CONT en el teclado y el **programa prosigue a partir del punto en que se detuvo.** Este comando permite continuar el programa **sin cambiar nada en las distintas operaciones y sin modificar las variables.**

• Digamos ahora que **la ejecución de un programa puede detenerse** de dos maneras distintas:

En modo directo (pulsando, por ejemplo, la tecla BREAK).

En modo programación (insertando en el programa la instrucción STOP).

DELETE

Este comando borra una o varias líneas del programa.

• Hay que advertir que **se puede suprimir una línea de un programa de varias maneras diferentes.** La manera más sencilla consiste en pulsar el número de la línea. Así, si mi programa contiene las líneas 1Ø, 2Ø, 3Ø, 4Ø y yo tecleo 2Ø (naturalmente, pulsando también la tecla RETURN o ENTER), suprimo la línea 2Ø y el programa sólo contendrá las líneas 1Ø, 3Ø, 4Ø.

También puedo —sobre todo si deseo borrar varias líneas— utilizar el comando DELETE.

Ejemplo: DELETE 2Ø-1ØØ. Este comando borrará del programa las línes 2Ø a 1ØØ.

EDIT

Se trata de **un comando de ayuda a la programación** (por tanto no es, en rigor, una instrucción del lenguaje Basic) del que puede prescindirse, pero cuya utilización es agradable. En efecto,

permite corregir errores de programación sin demasiada dificultad.

Sin embargo, este comando, cuya presencia en los ordenadores es bastante frecuente, coloca al ordenador en **modo de edición,** es decir, que el usuario puede modificar fácilmente el contenido de una línea errónea sin tener que volver a escribir todo el texto en la pantalla. Ya hemos explicado antes la diferencia entre un editor de textos y un tratamiento de textos; por tanto, no insistiremos más en estas modalidades.

El comando EDIT está muy lejos de tener las mismas posibilidades en todos los sistemas, por lo que conviene que usted se atenga a las instrucciones del manual de utilización de su ordenador.

LIST

Este comando permite visualizar en la pantalla una o más líneas del programa que acaba de introducirse mediante el teclado, o que se ha «conservado» a partir de una memoria auxiliar.

Cuando se escribe un programa (y se añaden líneas de programación en orden anárquico) es necesario muchas veces utilizar este comando, ya que presenta visualmente las líneas del programa en un orden lógico (creciente).

• Naturalmente, puede presentar visualmente **una sola línea del programa** (ejemplo LIST 5Ø: presenta la línea 5Ø), **todas las líneas del programa** (LIST) o **únicamente las líneas deseadas** (LIST 3Ø-1ØØ: presenta las líneas desde la 3Ø a la 1ØØ).

• **Si se desea imprimir las listas** en papel, en muchos ordenadores basta transformar LIST en LLIST.

NEW

Este comando hay que utilizarlo con precaución.

En efecto, **borra todo el programa Basic** que hay en memoria interna y reinicializa las variables a cero.

RUN

Es el comando de ejecución del programa

• **RUN sin argumento** provoca la ejecución a partir de la primera línea del programa.

• **RUN con un argumento** (por ejemplo, RUN 5Ø) provoca la ejecución del programa a partir de la línea correspondiente al argumento (en el ejemplo anterior: a partir de la línea 5Ø).

Documente sus programas

Los diez comandos anteriormente reseñados son los más empleados. En el capítulo siguiente veremos que algunas declaraciones pueden también emplearse en modo directo, es decir, como comandos (por ejemplo PRINT). Antes de analizar las principales instrucciones del lenguaje de programación Basic, detengámonos en la declaración REM, que sólo sirve... ¡para ayudar al programador y sus lectores!

REM

El lenguaje Basic le ofrece la posibilidad de **colocar comentarios u observaciones en su programa** utilizando la instrucción REM. Empléela tantas veces como pueda (excepto si le falta sitio en memoria central).

• **El programa no tiene en cuenta las líneas que empiezan por** REM; para él, es como si no existieran, pero para el que deba leer el programa ¡es como pan bendito!

• Ejemplo:
1Ø REM programa de prueba
...
5Ø REM la línea siguiente calcula el valor...

• En muchos Basic, REM queda abreviado en '
1Ø ' programa de prueba.

Las instrucciones

PRINT

La instrucción PRINT ya nos es familiar, puesto que es una de las primeras que hemos empleado en ocasión de crear nuestro primer pequeño programa («*Spot* publicitario»).

La comisión principal de esta instrucción es **presentar en pantalla los datos o los resultados.**

La instrucción PRINT puede utilizarse en modo directo o en modo de programación.

• **Modo directo (comando)**
El ordenador se utiliza, en este caso, como una simple calculadora.

Ejemplo:
PRINT 4 + 12
16
La respuesta del ordenador es inmediata.

Este modo es utilizable no sólo para presentar visualmente los resultados de un cálculo, sino también para la visualización del lugar en memoria: PRINT MEM (este comando no existe en todos los Basic) o también *el valor de una variable*. Una variable es una casilla de memoria cuyo contenido es variable; se identifica por medio de una o varias letras. Al principio, y mientras no hay programa en el ordenador, todas las casillas están a cero.

Ejemplos:
PRINT A
Ø
PRINT D
Ø
Mientras se está realizando un programa, se podrá emplear el comando PRINT para visualizar el contenido de las variables.

• **Modo programación**
En modo programación, este comando permite también visualizar el contenido de las variables o de un mensaje cualquiera escrito entre comillas.

Ejemplo: 1Ø PRINT «Esto es un mensaje»

La instrucción PRINT ordena asimismo *el paso a la línea siguiente:* por tanto, se empleará para *crear espacios entre las líneas.*

Ejemplo: 10 PRINT «Esto es un mensaje»
20 PRINT
30 PRINT
40 PRINT
50 PRINT «destinado a usted»

Ejecutado (RUN) este pequeño programa, dará en pantalla la imagen siguiente:

```
Esto es un mensaje



destinado a usted
```

Digamos también que es posible *posicionar el mensaje* en un lugar cualquiera de la pantalla o de la hoja de papel empleando la instrucción TAB (que funciona como el tabulador de una máquina de escribir) o darle un formato especial empleando la instrucción PRINT USING. Estas dos últimas instrucciones son algo sofisticadas y no las trataremos en este pequeño manual de iniciación.

Finalmente hay que advertir que PRINT emplea a menudo la *coma o el punto y coma.*

Si el *punto y coma* se coloca detrás de PRINT, el paso a la línea siguiente no se produce.

Ejemplo: 10 PRINT «Yo estoy; (; impide el retorno del carro)
20 PRINT « »; (creación de un blanco)
30 PRINT «contento»

La ejecución de este programa dará en pantalla:

```
Yo estoy contento
```

Si se pone *una coma* (ver el párrafo dedicado a los operadores), los resultados quedan espaciados en la pantalla (la coma, en efecto, hace las veces de una tabulación equivalente a 16 columnas).

Si lo que se desea obtener no es una presentación en pantalla, sino la *impresión* sobre papel, no se empleará PRINT, sino LPRINT.

PRINT se abrevia algunas veces en?
Ejemplo: ? 3* 4
12

- **Síntesis de la instrucción PRINT**

1 Cada instrucción PRINT provoca el salto a la línea siguiente, excepto si PRINT termina por:
una coma (,): provoca una tabulación
un punto y coma (;): provoca el enlace de los mensajes.

2. Los caracteres que PRINT ha de presentar visualmente estarán siempre entre comillas.

3. PRINT presenta visualmente los valores de las variables (numéricas o alfanuméricas).

4. PRINT es utilizable en modo directo o en modo programación.

5. La instrucción PRINT escrita sola provoca el salto a la línea siguiente.

6. La introducción PRINT puede completarse con TAB (para obtener una tabulación) o USING (para obtener una compaginación).

7. PRINT se abrevia a veces en?

LET

Esta instrucción (facultativa en numerosos Basic) **asigna un valor a una variable.** Muchas veces tiene el mismo valor que el signo «=» en una ecuación algebraica.

Así, las dos líneas siguientes tienen el mismo significado:

10 A = 17
10 LET A = 17

LET parece, pues, tener un doble empleo. Sin embargo, hay que hacer observar que muchas expresiones de asignación no tienen sentido alguno en matemática y, por consiguiente, LET toma todo su valor de instrucción de asignación.

Ejemplo: 10 LET B = B + 1

Esta línea, que no tiene ningún sentido en álgebra, significa que hay que añadir 1 al valor numérico de B y tener en cuenta este nuevo valor. En Basic es frecuente hallar líneas similares.

Digamos también que es el valor situado a la derecha del signo el que está asignado a la variable situada a la izquierda, y no al contrario.

Ejemplo de programa que utiliza LET:

```
10 LET A = 17
20 LET B = 33
30 LET C = A + B
40 LET D = A - B
50 LET D = A * B
60 PRINT A, B, C, D, E

RUN
17      33      50      —16      561
```

• **Observación**

Para obtener la presentación visual de los resultados de las variables en una sola línea, las variables a presentar se separan mediante comas (línea 60).

INPUT

A usted le gustaría interrogar al ordenador, cosa muy razonable, ya que su aparato está hecho para esto. Sin embargo, déle también a él la ocasión de interrogarle a usted de vez en cuando. Si en el programa coloca la instrucción INPUT, **el ordenador se detendrá, presentará un signo de interrogación y esperará su respuesta.** La instrucción INPUT es, pues, un excelente medio para entrar informaciones en el ordenador... sin importar cuáles sean estas informaciones.

• **La instrucción** INPUT **coloca el valor que usted acaba de entrar en una variable;** ahora bien, sabemos que existen variables numéricas y variables alfanuméricas, por lo que el INPUT habrá de indicar muy bien el tipo de variable que espera.

Ejemplo: INPUT A (el ordenador espera una variable numérica)
INPUT A$ (el ordenador espera una variable alfanumérica)

• **Ejemplo de programa** (para calcular la circunferencia de un círculo de radio variable).

Le recordaremos (por si hubiera olvidado su geometría) que la longitud de una circunferencia se obtiene mediante la fórmula siguiente: Longitud = $2 \times \pi \times$ Radio.

He aquí nuestro pequeño programa:


```
10 INPUT R
20 LET C = 2*3.1416*R
30 PRINT «ÉSTA ES LA LONGITUD DE LA CIRCUNFERENCIA»
40 PRINT C
```

• **Observación:** en lenguaje Basic, se utiliza el símbolo * **para multiplicar.**

Si deseamos introducir una variable alfanumérica, basta añadirla al programa: INPUT A$.

Inmediatamente se ve que si el ordenador presenta un signo de interrogación cuando espera una respuesta numérica y el mismo signo de interrogación cuando espera una respuesta alfanumérica, existe un gran riesgo de confusión y de bloqueo del programa (cuando la respuesta a la pregunta no es correcta, el ordenador presenta un mensaje de error y algunas veces se «planta»).

• **¿Qué hacer para que el signo de interrogación sea significativo?** Es muy sencillo: colocar un mensaje a su izquierda. Basic ha pensado en ello y la instrucción INPUT se comporta como la instrucción PRINT: basta colocar una cadena de caracteres entre comillas para que esta cadena aparezca en pantalla al mismo tiempo que el signo de interrogación.

Observemos —es muy importante— que, entonces, **es indispensable colocar un punto y coma** (;) detrás de la cadena alfanumérica.

• **Volvamos al mismo ejemplo, pero colocando un mensaje.**

```
10 INPUT «Cuál es su nombre»; A$
20 INPUT «Cuál es el radio de la circunferencia»; R
30 LET C = 2*3.1416*R
40 PRINT «Querido  »
45 PRINT A$
50 PRINT «Esto es la longitud de la circunferencia»
55 PRINT C
```

Evidentemente, nuestro programa es didáctico, pero muy poco elegante (podría escribirse con muchas menos líneas).

• Finalmente, conviene advertir que no hemos puesto signo de interrogación a nuestras preguntas, ya que —recordémoslo— **la instrucción** INPUT **coloca de oficio este signo de interrogación.**

GOTO

Si bien no es recomendable utilizar con frecuencia esta instrucción (so pena de no poder releer convenientemente un programa), no debemos ocultar el placer que procura a todos los jóvenes programadores que gustan de su uso y, a veces, abuso.

Esta instrucción, que ya hemos encontrado en nuestra presentación del «*spot* publicitario», **provoca un salto a otra línea que no es la normalmente ejecutada.** En efecto, si las instrucciones de un programa se ejecutan habitualmente en el orden de su numeración lógica, la presencia de un GOTO altera esta orden y provoca un salto incondicional a otra línea.

• La instrucción GOTO raramente se emplea sola, ya que, en este caso, provoca simplemente un salto a otra línea de programa. **Lo más corriente es que se utilice con los operadores de comparación** (que comparan un valor con otro y emplean los símbolos igual (=), mayor que (>), menor que (<), etc.) y la instrucción IF. Esta instrucción, que más adelante estudiaremos, significa *si*.

• **Imaginemos un pequeño programa** que utilice GOTO, el operador «=» e IF.

```
10 INPUT «Pulse en el teclado, a voluntad, 1 o 2»; X
20 IF X = 1 GOTO 50
30 PRINT «Ha pulsado 2»
30 GOTO 10
50 PRINT «Ha pulsado 1»
60 GOTO 10
```

Si pulsa 1, el programa salta la línea siguiente y presenta en pantalla «Ha pulsado 1». El GOTO le obliga a volver a la línea 10 y el programa se reanuda. Si ahora pulsa 2, el programa prosigue su marcha habitual, pasa a la línea 30, y presenta en pantalla «Ha pulsado 2». El GOTO de la línea siguiente provoca un nuevo salto a la línea 10 y el programa se reanuda.

Evidentemente, este programa adolece de numerosos defectos..., que ya aprenderemos a corregir. Así, si pulsa 3, el programa presenta también «Ha pulsado 2». Esto se debe a que la comprobación de la línea 20 sólo provoca el salto hacia la línea 50 cuando usted pulsa 1. Por tanto, será necesario impedir al teclado que no acepte nada excepto 1 ó 2, es decir, prever en el programa que toda instrucción que no sea 1 ó 2 provoca el salto hacia otra línea. Es lo que vamos a hacer (las nuevas instrucciones están en las líneas 25, 26 y 27).


```
10 INPUT «Pulse en el teclado, a voluntad, 1 o 2»; X
20 IF X = 1 GOTO 50
25 IF X = 2 GOTO 30
26 PRINT «Ha pulsado otra cifra. Vuelva a empezar»
27 GOTO 10
30 PRINT «Ha pulsado 2»
40 GOTO 10
50 PRINT «Ha pulsado 1»
60 GOTO 10
```

Naturalmente hay maneras más elegantes de hacer este pequeño programa, pero para ello deberíamos utilizar instrucciones y operadores que todavía no hemos estudiado.

De cualquier forma, este pequeño programa (que puede usted «adornar» a su gusto) le muestra ya las enormes posibilidades del comando GOTO.

IF THEN

Ya hemos encontrado IF cuando hemos estudiado GOTO.

IF se utiliza en muchas instrucciones. Si se emplea con THEN en la declaración IF THEN, **realiza un salto condicional,** a diferencia de GOTO, que realiza un salto incondicional.

• **SI la condición se cumple,** ENTONCES **el ordenador efectúa el salto a la línea especificada.** La potencia de IF THEN puede ser decuplicada mediante la utilización de operadores de comparación. Es posible imaginar numerosas situaciones:

— si... es mayor que... entonces...
— si... es menor que... entonces...
— si... es diferente de... entonces...
etcétera...

• **Volvamos a nuestro programa anterior** y modifiquémoslo para utilizar IF THEN:

```
10 INPUT «Pulse en el teclado, a voluntad, 1 o 2»; X
20 IF X = 1 THEN 50
25 IF X = 2 THEN 30
26 PRINT «Ha pulsado otra cifra. Vuelva a empezar»
27 GOTO 10
30 PRINT «Ha pulsado 2»
40 GOTO 10
50 «Ha pulsado 1»
60 GOTO 10
```

Puede comprobarse que es posible sustituir el GOTO de las líneas 20 y 25, pero no así los demás GOTO (pruebe y verá: el ordenador presentará un mensaje de error).

• **En un salto incondicional, sólo está permitido** GOTO.

• **En un salto condicional se puede utilizar** GOTO **o** THEN.

IF... THEN... ELSE

Del inglés: *si... entonces... si no*

Se trata de una instrucción complementaria a la instrucción IF THEN, que indica al programa el camino a seguir si la condición especificada por IF THEN no se cumple. Es perfectamente posible prescindir de esta instrucción, aunque facilita enormemente la escritura del programa.

ELSE será simplemente utilizado en nuestros programas de aplicación.

Ejemplo:

```
10 INPUT «Entre un número igual o inferior a cinco»; X
20 IF X > 5 THEN GOTO 10 ELSE PRINT X
```

GOSUB RETURN

La instrucción GOSUB es muy empleada cuando un programa puede dividirse en varios pequeños programas. Se tendrá entonces el programa principal y programas accesorios que efectúan las operaciones repetitivas. Estos programas accesorios se denominan subprogramas o subrutinas (en inglés = subroutine). **Un subprograma es un pequeño programa ejecutable a partir de cualquier punto de un programa.** El empleo de subprogramas hace que la escritura de un programa sea mucho más clara, más rápida y más legible.

Así pues, si para ejecutar correctamente un programa tenemos normalmente necesidad de un cálculo bastante complejo, sería lástima tener que escribir cada vez las instrucciones necesarias para este cálculo, cuando, en realidad, *basta crear un bloque conteniendo el programa de cálculo y remitir regularmente el programa a este bloque.* Por tanto, un subprograma está constituido por tres partes:

— **la entrada** en el subprograma (se suele utilizar REM para indicar el objeto de este subprograma),

— **el cuerpo** del programa (las instrucciones necesarias para la ejecución del tratamiento de los datos),

— **el final** del programa (es decir, el retorno a la línea anterior al salto).

Esto puede realizarse gracias a la instrucción GOSUB (comando de salto al subprograma) y a la instrucción RETURN que obliga al programa a volver al estado anterior.

• **Pueden crearse tantas subrutinas como se deseen, mientras cada una de ellas termine con la instrucción RETURN.**

• **Ejemplo:**

```
10 INPUT «Pulse un número cualquiera inferior a 1000»;X
20 GOSUB 290
30 PRINT «Esta subrutina tenía por función elevar al cuadrado
un número cualquiera»
40 END
290 REM SUBRUTINA DE ELEVACIÓN AL CUADRADO
300 PRINT «Ha pulsado la cifra    »;X
310 LET X=X*X
320 PRINT «El cuadrado de esta cifra es    »; X
330 RETURN
```

• La instrucción END indica una detención del programa. La presentaremos más tarde.

• Esta subrutina (comprendida entre las líneas 290 y 330) merece verse con más detalle.

— La primera línea (290) es una línea de observación, que permitirá, en el momento de la relectura de un programa, saber inmediatamente en qué línea empieza una subrutina particular.

— La segunda línea (300) es una indicación de la cifra pulsada.

— En la tercera línea (310), la cifra es elevada al cuadrado.

— En la cuarta línea (320) se presenta visualmente el resultado de la operación.

— La quinta línea provoca el retorno a la situación anterior, es decir, después del GOSUB, y el programa prosigue su curso normal.

De esta manera es posible crear numerosas subrutinas con diversas funciones (cálculo de cosenos, selección, verificación de las entradas, etc.) y emplearlas únicamente en caso de necesidad.

FOR TO NEXT

La instrucción FOR TO NEXT obliga al programa a ejecutar una instrucción tantas veces como se especifique: es lo que se llama **un bucle de programación.**

Ya hemos hablado de la potencia de FORT TO NEXT en uno de nuestros pequeños programas («*Spot* publicitario»).

FOR TO NEXT no es el único bucle de programación; además de él se utiliza muy a menudo GO TO e IF THEN.

El pequeño programa que sigue nos mostrará cómo funciona este bucle:

```
10 FOR A = 1 TO 9
20 PRINT A
30 NEXT A
40 END

RUN
1
2
3
4
5
6
7
8
9
```

— **La primera línea** del programa especifica que la variable A tomará sucesivamente los valores de 1 a 9.

— **La segunda línea** pide la impresión de los valores sucesivos.

— **La tercera línea** provoca el retorno a la primera línea mientras el bucle no ha terminado (es decir, mientras la variable A no ha alcanzado el valor 9).

• **Aplicación: el código ASCII**

Para los que tienen ante sí un ordenador, proponemos el pequeño programa siguiente, que presenta en pantalla el valor del código ASCII. Es el momento de decir algunas palabras sobre este código propuesto por el *American Standard Code for Information Interchange.*

Los caracteres alfanuméricos, como los signos especiales y algunas funciones del ordenador (por ejemplo, accionar un sistema de sonido interno), están codificados. Como el lenguaje informático es binario y los ordenadores suelen ser de 8 bits, se dispone por cada byte de 256 posibilidades distintas (2 elevado a 8).

Para obtener la presentación visual del código correspondiente a un valor (por ejemplo, 65), basta pulsar en el teclado PRINT CHR$ (65) (del inglés CHaRacter) y el ordenador responde A. El código ASCII (se pronuncia aski) de A es 65.

Como la codificación de las letras y las cifras no precisa la totalidad de los códigos disponibles, los códigos libres se destinan a funciones precisas, a grafismos, a colores, etc.

Hemos de advertir que como el español utiliza algunos caracteres (signos de principio de admiración e interrogación, letra ñ, etcétera) no previstos en el código ASCII, algunas veces pueden surgir conflictos entre el ordenador y sus periféricos.

• **La clasificación de los códigos es la siguiente:**

Códigos de 0 a 31

Estos códigos ya no tienen actualmente valor estándar: sirven para enviar mensajes a la pantalla o a otros periféricos.

Códigos 32 a 90

Son idénticos en todos los ordenadores y están asignados a los signos especiales y a las letras mayúsculas.

Códigos 91 a 96

Varían generalmente de un ordenador a otro y están asignados a los desplazamientos del cursor o a los acentos.

Códigos 97 a 122

Están asignados a las letras minúsculas.

Códigos 123 a 256

Estos códigos varían según los ordenadores y pueden estar asignados a los caracteres especiales, etc.

• Si deseamos ver **cómo utiliza nuestro ordenador los códigos** ASCII, basta ejecutar el programa siguiente, que utiliza en dos ocasiones el bucle FOR TO NEXT:

```
10 FOR N = 33 TO 132
20 PRINT «CODE ASCII» ;N;
21 PRINT «=»;
30 FOR X = 1 TO 200
35 NEXT X
40 PRINT CHR$ (N);
60 NEXT N
```

Nota: si dispone de una impresora, suprima el bucle de la línea 30 y sustituya PRINT por LPRINT.

```
ODE ASCII 33 =! CODE ASCII 34 =" CODE ASCII 35 =# CODE ASCII 36 =$ CO
 =( CODE ASCII 41 =) CODE ASCII 42 =* CODE ASCII 43 =+ CODE ASCII 44
II 48 =0 CODE ASCII 49 =1 CODE ASCII 50 =2 CODE ASCII 51 =3 CODE ASC
 ASCII 56 =8 CODE ASCII 57 =9 CODE ASCII 58 =: CODE ASCII 59 =; CODE
CODE ASCII 64 =@ CODE ASCII 65 =A CODE ASCII 66 =B CODE ASCII 67 =C
 =G CODE ASCII 72 =H CODE ASCII 73 =I CODE ASCII 74 =J CODE ASCII 75
II 79 =O CODE ASCII 80 =P CODE ASCII 81 =Q CODE ASCII 82 =R CODE ASC
 ASCII 87 =W CODE ASCII 88 =X CODE ASCII 89 =Y CODE ASCII 90 =Z CODE
CODE ASCII 95 =_ CODE ASCII 96 =@ CODE ASCII 97 =A CODE ASCII 98 =B
 102 =F CODE ASCII 103 =G CODE ASCII 104 =H CODE ASCII 105 =I CODE AS
 =M CODE ASCII 110 =N CODE ASCII 111 =O CODE ASCII 112 =P CODE ASCI
```

• Comentarios e ilustración

Este programa sólo tiene en cuenta los caracteres ASCII comprendidos entre 33 y 132, es decir, los caracteres ASCII usuales. Si desea reproducir el efecto de los caracteres Ø a 32 y 133 a 256, le aconsejamos que no utilice el comando para la impresora LPRINT y limítese a asistir a un espectáculo imprevisible en la pantalla. Sin embargo, nada tema: ¡el ordenador no explotará!

Por lo que respecta al programa propiamente dicho, hemos de hacer una observación a cuanto se ha comentado:

— línea 2Ø: el «;» después de «CODE ASCII» y después de N sirve sólo para impedir un salto de línea y para presentar visualmente los resultados en una sola línea (en la medida de lo posible). La misma observación puede hacerse para las líneas 21 y 4Ø.

Nada le impide eliminar este punto y coma...

READ DATA/RESTORE

La primera introducción podría traducirse por: LEE DATOS; la segunda por: RESTITUYE (esos datos).

Un ordenador se nutre de datos. Éstos pueden entrarse en el teclado por mediación de la introducción INPUT (que no existía en el primer Basic de Dartmouth), pero también pueden estar ya contenidos en el programa. Ya hemos visto la manera de entrar los datos por medio de la instrucción INPUT. Veamos ahora **el modo de leer estos datos contenidos en un programa.**

• La instrucción READ lee los datos contenidos en una línea DATA. Por tanto, la instrucción READ ha de figurar al principio del programa, mientras que la instrucción DATA se encuentra habitualmente al final del programa (pero siempre después de READ).

• ¿Qué hay que poner en la instrucción DATA? Todo lo que quiera: tanto cifras como letras.

• Para leer y utilizar estos datos, el ordenador los asignará a variables. Ahora bien, como ya sabemos, el ordenador trata las variables numéricas y alfanuméricas de manera distinta (para las primeras, letras; para las segundas, letras acompañadas del símbolo del dólar: A$).

• Los datos contenidos en DATA han de estar separados por comas.

• No puede haber más READ que datos.

• Ejemplo:

```
10 REM Lectura de datos
20 READ A, B, C, D
30 PRINT A, B, C, D
40 DATA 1,2,3,4
```

```
10 REM Lectura de datos
20 READ A$, B$, C$, D$
30 PRINT A$, B$, C$, D$
40 DATA JULIO, JUAN, IRENE, ANA
```

En la línea 2Ø está la lectura de los datos contenidos en la línea 4Ø, luego retorno a la línea 3Ø y, finalmente, presentación visual de los resultados de la lectura.

• Si deseamos leer varias veces los datos, es indispensable utilizar la instrucción RESTORE. Prosigamos nuestro programa (sin utilizar RESTORE) y añadámosle una línea (25) pidiendo una segunda lectura de los datos:

```
10 REM Lectura de datos
20 READ A,B,C,D
25 READ A,B,C,D
30 PRINT A,B,C,D
40 DATA 1,2,3,4
```

Si ejecutamos este programa, la pantalla presentará un mensaje de error: hay más READ que datos. Cuando el ordenador ha leído una vez los datos, no vuelve atrás para una nueva lectura, excepto si se lo pedimos por medio de la instrucción RESTORE (que sustituye una de las instrucciones READ):

```
10 REM Lectura de datos
20 READ A,B,C,D
22 PRINT A,B,C,D
25 RESTORE
30 PRINT A,B,C,D
40 DATA 1,2,3,4
```

El lector hallará numerosas aplicaciones de READ DATA en los programas de aplicación propuestos en este manual.

END/STOP

Esta instrucción indica el final de un programa o de un subprograma. En algunos Basic ha de estar colocada obligatoriamente al final del programa; en otros, su utilización es libre. Algunos dialectos aceptan el END en cualquier lugar del programa; otros, lo rechazan.

• **Para señalar una detención provisional** (y conservar los valores de las variables intactos) **se emplea el comando** STOP. En efecto, para proseguir un programa después de END es necesario utilizar RUN y las variables quedan nuevamente inicializadas a cero, mientras que **para proseguir un programa después de** STOP, **basta pulsar** CONT **y el programa prosigue sin modificaciones.**

Cabría preguntarse de qué puede servir indicar el final de un programa o de un subprograma. En realidad, esta instrucción demostrará ser muy útil cuando se crean operaciones de salto (por ejemplo GOTO) y es indispensable terminar el programa en un punto preciso.

• **Ejemplo:**
Prosigamos un programa anterior y modifiquémoslo:

```
10 INPUT «Pulse en el teclado, a voluntad, 1 o 2» ;X
20 IF X=1 GOTO 50
30 PRINT «Ha pulsado 2»
40 REM
50 PRINT «Ha pulsado 1»
60 REM
```

Las líneas 40 y 60 se han suprimido (para no suprimirlas totalmente se ha colocado la instrucción REM).

```
RUN
Pulse en el teclado, a voluntad, 1 o 2? 1
Ha pulsado 1
```

Perfectamente: todo va bien. Prosigo.

```
RUN
Pulse en el teclado, a voluntad, 1 o 2? 2
Ha pulsado 2
Ha pulsado 1
```

Moraleja provisional: ¡el ordenador es un perfecto idiota!

En realidad, después de haber ejecutado convenientemente la línea 30, el programa pasa a la línea 40 y luego a la línea 50, que es una línea de salto condicional.

Moraleja: para que una línea de salto no parasite el programa, es necesario que vaya habitualmente precedida de la instrucción END.

• **Continuemos el programa corregido**

```
10 INPUT «Pulse en el teclado, a voluntad, 1 o 2»;X
20 IF X=1 GOTO 50
30 PRINT «Ha pulsado 2»
40 END
50 PRINT «Ha pulsado 1»
60 END
```

Ejecútelo y verá que todo funciona a las mil maravillas... (mientras no pulse más que 1 ó 2, pero ya hemos tratado este problema). Con esto puede ver que la instrucción END no es tan inútil como parece.

DIM

La instrucción DIM es uno de los comandos más utilizados en lenguaje Basic y, sin embargo, lo evitan muchos principiantes, ya que no comprenden exactamente cuál es su función. Por esto le dedicaremos algunas páginas.

Este comando sirve para reservar en la memoria del ordenador un lugar. Este lugar se empleará para almacenar en él tablas de variables.

Hemos visto ya que se podían almacenar variables empleando el READ y el DATA, pero este sistema sólo es aplicable cuando las variables están colocadas unas después de otras en un orden inmutable. Si tenemos necesidad de colocar las variables en un orden cualquiera, esta instrucción no servirá. Naturalmente, lo que sí podemos es emplear cada vez una variable distinta, pero esto sería poco económico.

DIM, según hemos dicho, sirve para reservar un lugar para las tablas variables.

• Pero, **¿qué es una tabla de variables?**
Para explicarlo, tomemos un ejemplo sencillo: supongamos una clase de 10 alumnos. El profesor de gimnasia, un enamorado de la informática, desea conservar las notas obtenidas por los

alumnos. Podría, claro está, decidir la utilización de una variable distinta para cada nota. Este método, poco elegante, no está, por otra parte, exento de peligros. En efecto, no tendría ningún problema para la clase mencionada, que sólo tiene 10 alumnos, pero si deseara conservar las notas de gimnasia de todos los alumnos de las escuelas donde enseña, no le bastarían todas las variables disponibles para llevar a cabo con éxito su trabajo. Por tanto, tendría que imaginar un método menos complicado en variables y, al mismo tiempo, más elegante. Este método existe en lenguaje Basic: con la utilización de la instrucción DIM se crean unas tablas.

Por tanto, en vez de emplear una variable distinta para cada alumno, sólo utilizará una variable única (ya que se trata, en realidad, de una categoría única: los puntos obtenidos) que permitirá al ordenador reconocer a cada alumno dándole un número, un índice.

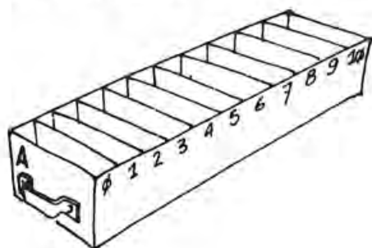
• **El ordenador reconocerá una variable de una tabla si se emplean los paréntesis después de la variable.**

De esta manera se obtiene una tabla de una sola variable, pero que contiene varios elementos. Si designamos a la variable A y a cada nota con un índice comprendido entre 1 y 10, obtendremos la tabla siguiente:

—nota del primer alumno	A (1)
—nota del segundo alumno	A (2)
—nota del tercer alumno	A (3)
—nota del cuarto alumno	A (4)
etc.	

• **Esta tabla es algo así como un armario clasificador con diversos compartimentos:** es fácil ver lo que contiene cada cajón, como es fácil también identificar cada cajón por una letra. El cajón puede dividirse, asimismo, en varios compartimentos, que también podrán numerarse.

Lo que acabamos de hacer es, sencillamente, identificar un cajón que contendrá las notas de gimnasia de los alumnos y dividir este cajón en varias partes.



El cajón A está dividido en 10 partes, lo que nos da 11 compartimentos distintos. Los compartimentos están numerados, y el primero lleva el número Ø. Este cajón contiene, pues, las notas de gimnasia de los 10 alumnos de la clase.

En informática, para crear este cajón se emplea la declaración DIM.

• Como puede verse, **esta instrucción reserva lugar en memoria** (el cajón está reservado a la puntuación de gimnasia y no contendrá más) y, por otra parte, **impone un límite al número de datos entrados** (el cajón sólo tiene 11 compartimentos; ni uno más).

• **Para formular esta instrucción** en Basic se escribirá, pues:

```
10 DIM A (10)
```

• **Observemos también que:**

— la instrucción ha de estar colocada **antes** de la utilización de la tabla y, por tanto —y con preferencia—, al principio del programa;

— A es el nombre de la variable, que se denominará **variable de lista.** Esta variable puede ser alfanumérica (por ejemplo, A$);

— **los paréntesis** permiten al ordenador distinguir una tabla de una variable;

— la dimensión de la variable (lo que está entre paréntesis), o **índice de la tabla,** puede ser una constante, el resultado de un cálculo o incluso una variable, a condición de que contenga un valor.

• **Ejemplo:**

```
10 CLS
20 DIM A (10)
30 FOR I = 1 TO 10
40 PRINT «Número del alumno»
50 PRINT I
60 INPUT «Introduzca su nota»; A (I)
70 NEXT I
80 END
```

Si se hace funcionar este programa mediante el comando RUN, obtendremos:

```
Número del alumno
1
```

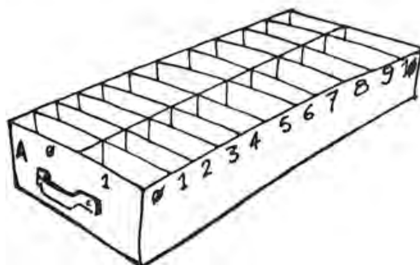
```
Introduzca su nota? 1Ø
Número del alumno
2
Introduzca su nota? 7
```

Digamos también que con la instrucción DIM (1Ø) se hubiera podido introducir el bucle FOR I = Ø TO 1Ø y, así, incluir 11 alumnos.

En principio —y salvo que la falta de espacio nos obligue a otra cosa— es preferible no utilizar el compartimento Ø.

Tablas de varias dimensiones

La tabla que hemos empleado sólo posee un cajón, una sola dimensión. A menudo es preciso emplear varios compartimentos en un cajón. Llegado el caso, **cada compartimento del cajón se identifica por dos coordenadas: el número de la línea y el número de la columna.** De esta manera podemos obtener un cajón dividido de la manera siguiente:



Este cajón representa **una variable de dos dimensiones.** Si volvemos al ejemplo anterior, podemos muy bien imaginar que el profesor de geografía, convencido por los resultados de su colega, decide introducir también en el ordenador las notas de sus alumnos. Como tenemos ya un compartimento (o variable) para las notas de gimnasia, bastará compartimentarlo en dos para que contenga también las notas de geografía. Esto se hace introduciendo una división transversal en el cajón (ver más adelante) y, en informática, añadiendo un segundo índice a la dimensión:

```
1Ø DIM A (1Ø,1)
```

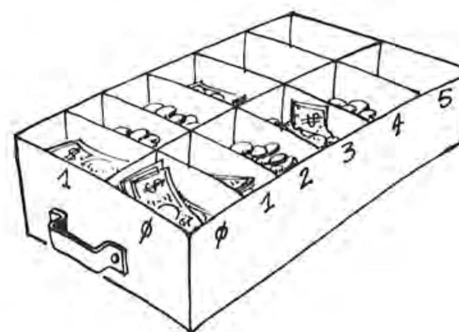
Puede observarse que ambos índices están separados por una

coma. Para utilizar esta tabla habrá que numerar los alumnos de 1 a 1Ø y las materias de Ø a 1.

Como las divisorias que separan los compartimentos están numeradas, es fácil hallar el contenido de una casilla o colocar en ella algo (piense en las casillas de los crucigramas).

• En informática, **para obtener el contenido de un casillero** se emplea la instrucción PRINT y **para indicar que es necesario colocar en él algún dato** se recurre a la instrucción LET (o, si no es obligatoria, al signo de asignación =).

• Ejemplo:



```
1Ø DIM A (1,5)
2Ø INPUT «Cuánto dinero en la hilera de la derecha,
                                  compartimento Ø »; A (Ø ,Ø )
3Ø INPUT «                        compartimento 1»; A (Ø ,1)
4Ø INPUT «                        compartimento 2»; A (Ø ,2)
5Ø INPUT «                        compartimento 3»; A (Ø ,3)
6Ø INPUT «                        compartimento 4»; A (Ø ,4)
7Ø INPUT «                        compartimento 5»; A«Ø ,5)
8Ø INPUT «Cuánto dinero en la hilera de la izquierda,
                                  compartimento Ø »; A (1,Ø )
9Ø INPUT «                        compartimento 1»; A (1,1)
1ØØ INPUT «                       compartimento 2»; A (1,2)
11Ø INPUT «                       compartimento 3»; A (1,3)
12Ø INPUT «                       compartimento 4»; A (1,4)
13Ø INPUT «                       compartimento 5»; A (1,5)
```

El pequeño programa que acabamos de reproducir ilustra una manera de emplear la declaración DIM para una tabla de dos dimensiones (cajón-compartimento con dinero).

• Para acabar, digamos que un tablero de una sola dimensión

se denomina **vector** y que se utilizará el término **matriz** para definir una tabla de dos o más elementos. En efecto, pueden crearse tableros que contengan tantos índices como se desee, pero su empleo resulta bastante complejo.

• *Obsérvese que:*

A	es una variable numérica
A (10)	es una tabla
A$	es una variable alfanumérica

El ordenador nunca confundirá estas tres variables.

Tablas de cadenas

Hemos visto anteriormente que la variable puede también ser alfanumérica (por ejemplo, A$). Para dar un ejemplo de tabla de cadenas recurriremos a los meses del año. Este ejemplo se emplea frecuentemente en los programas.

• Coloquemos **los doce meses del año en una tabla** (esta tabla tendrá usos múltiples, como convertir los meses expresados en cifras, en letras, en la tabla de los biorritmos, etc.). Creamos el programa siguiente:

```
10 DIM M$ (12)
```

donde M$ es una variable alfanumérica de una tabla que contiene doce (en realidad trece) elementos, que son los meses del año:

```
 10 DIM M$ (12)
 20 M$  (1) = «Enero»
 30 M$  (2) = «Febrero»
 40 M$  (3) = «Marzo»
 50 M$  (4) = «Abril»
 60 M$  (5) = «Mayo»
 70 M$  (6) = «Junio»
 80 M$  (7) = «Julio»
 90 M$  (8) = «Agosto»
100 M$  (9) = «Septiembre»
110 M$ (10) = «Octubre»
120 M$ (11) = «Noviembre»
130 M$ (12) = «Diciembre»
```

Esta tabla permite múltiples utilizaciones, que usted puede divertirse en imaginar.

Los operadores

Un operador es, simplemente, **un símbolo que representa una operación a efectuar.** Desde el momento en que se aprende a calcular ya se empiezan a utilizar operadores como, por ejemplo, el signo + de uso tan corriente.

Los operadores empleados en lenguaje BASIC suelen ser símbolos idénticos a los que se usan en matemáticas; algunas veces —raramente— son palabras.

Los operadores pueden clasificarse en cuatro categorías distintas:

1 — Los operadores matemáticos
2 — Los operadores lógicos
3 — Los operadores de comparación
4 — Los operadores de función

Los operadores matemáticos

Son los operadores que nos resultan más familiares.
Los principales son:

+	utilizado para la suma
−	utilizado para la resta
*	utilizado para la multiplicación (no ha de

	emplearse nunca «x», como en aritmética)
/	utilizado para la división (no ha de emplearse «:», como en aritmética)
^ o ↑	para la elevación a potencias (el signo utilizado puede variar de un Basic a otro).

Es importante destacar que estos operadores pueden relacionar entre sí no solamente cifras (como en aritmética), sino también constantes (numéricas o alfanuméricas), variables (numéricas o alfanuméricas) e incluso funciones matemáticas.

• **Ejemplo:** el pequeño programa que sigue muestra una utilización de los operadores «+» y «−»:

```
10 LET A = 9
20 LET B = 31
30 LET C = A+B
40 PRINT C
```

Ejecución:

```
RUN
40
```

El operador «+» ha sumado las variables numéricas A y B.

• Cuando el operador «+» no suma cifras sino cadenas de caracteres, no se habla ya de suma, sino de **concatenación.**

• Digamos asimismo que el signo «=» se emplea de la misma manera que en matemáticas (signo de igualdad), pero también para colocar una variable en una zona determinada de la memoria: es lo que se llama una asignación, y el símbolo «=» es **un símbolo de asignación.**

Los operadores lógicos

Se denominan también **operadores booleanos** (por el nombre de George Boole, inventor de una álgebra lógica). Estos operadores emplean palabras inglesas. Los principales son:

AND	empleado para la operación equivalente a **Y**
OR	empleado para una operación equivalente a **O**
NOT	empleado para una operación equivalente a **NO.**

Estos operadores son muy empleados **para operaciones lógicas** que intervienen en los programas. Por ejemplo, se pide al usuario que responda a dos preguntas, seleccionando la cifra 1 si está de acuerdo y la cifra 2 si no lo está. De esta manera, se podrá pedir al ordenador la ejecución del programa solamente si una de las tres condiciones que se expresan a continuación se cumple (a elegir):
— si la respuesta a las dos preguntas es 1 (empleando el operador AND)
— si la respuesta a una de las preguntas es 1 (empleando el operador OR)
— si la respuesta a ambas preguntas no es 1 (empleando el operador NOT).

Volveremos a hallar estos operadores en los programas de aplicación.

Los operadores de comparación

Estos operadores (o **símbolos relacionales**) son también muy empleados en matemáticas. En lenguaje Basic se utilizan como **operadores de comparación numérica** (como en matemáticas), pero también como **operadores de comparación alfanumérica.** Para comparar entre sí cadenas de caracteres alfanuméricos, el ordenador transformará las letras en su valor cifrado convencional (= código ASCII, ver diccionario).

• **Los principales operadores de comparación son:**

Símbolo	**Función numérica**	**Función alfanumérica**
=	igual a	igual a
>	mayor que	sigue
<	menor que	precede
<=	menor o igual a	precede o es igual a
>=	mayor o igual a	sigue o es igual a
<>	distinto de	distinto de

En el programa que ilustra la construcción de un diagrama de flujo (ver página 105) se hallará un ejemplo de utilización de estos operadores relacionales (con la instrucción IF... THEN).

Los operadores de función

Generalmente, estos operadores sirven para **indicar al ordenador que ha de efectuar una u otra función.** Intervienen en la sintaxis de la frase y es absolutamente indispensable emplearlos bien, so pena de obtener un programa anárquico.

Símbolo	Nombre
.	Punto
;	Punto y coma
!	Signo de admiración
:	Dos puntos
"	Comillas
,	Coma
()	Paréntesis
$	Dólar
#	Sostenido
%	Por ciento
\	Barra invertida
?	Signo de interrogación
'	Apóstrofo

- **Descripción de los operadores de funciones**

Ya hemos dicho que estos operadores tienen múltiples funciones, por lo que hay quien prefiere distinguir los operadores que sirven para las entradas/salidas (por ejemplo: visualización en una sola línea, etc.) y los operadores que declaran tipos (por ejemplo: en doble precisión, etc.).

Aquí nos limitaremos a precisar la función de cada uno de los operadores más arriba mencionados.

Punto

El punto sirve, principalmente, para **indicar los decimales.** En efecto, tiene el mismo valor que la coma, tal como la utilizamos en Europa. Por tanto, no se escribirá 7,65 sino **7.65**

Digamos también que ciertos dialectos emplean el punto en muchos casos: para indicar la última línea de un programa, la abreviatura de un comando o de una declaración (ejemplo: L. por LIST), etc.

Punto y coma

Lo más frecuente es que el punto y coma se utilice con la instrucción PRINT. En este caso, **sirve para colocar varios elementos en una misma línea** e impide el paso a la línea siguiente.

- **Ejemplo:**

```
10 INPUT A$
20 INPUT B$
30 PRINT A$;B$
```

Las dos variables alfanuméricas A$ y B$ se yuxtapondrán en la misma línea.

- **Ejemplo:**

```
10 PRINT «Editorial Juventud le desea  »;
20 PRINT «una excelente lectura»
RUN
«Editorial Juventud le desea una excelente lectura»
```

La frase está colocada en una sola línea. Observe que es necesario dejar un blanco al final de la primera línea, antes de cerrar la comillas, para espaciar las dos líneas.

Signo de admiración

Este signo se utiliza para indicar que **una variable está en «simple precisión»** (generalmente seis cifras después del punto: 0.333333).

Como al principio todas las variables están en simple precisión, este operador sólo se emplea en el caso en que se precise que una variable está en «doble precisión» (generalmente dieciséis cifras después del punto: 0.3333333333333333).

Dos puntos

Este símbolo se emplea cuando se desea **escribir varias declaraciones en una misma línea.** En este libro hemos decidido utilizar una sola declaración por línea (como en el Basic estándar), pero generalmente hubiéramos podido escribir nuestros programas de manera más «condensada». La utilización de los dos puntos hace que pueda economizarse espacio en la memoria, pero este empleo ha de ser muy prudente, sobre todo con instrucciones de salto (GOTO, GOSUB, etc.).

Hemos de advertir que ciertos ordenadores emplean la barra invertida (\) en vez de (:).

• **Ejemplo:**

```
10 PRINT «BRAVO»
20 FOR X = 1 TO 200
30 NEXT X
40 GOTO 10
```

Este pequeño programa puede escribirse en una sola línea:

```
10 PRINT «BRAVO»: FOR X = 1 TO 200: NEXT X: GOTO 10
```

Comillas

Las comillas desempeñan una función muy importante: **delimitan cadenas de caracteres e indican cómo han de tratarse.**

Así, en la instrucción PRINT, lo que está colocado entre comillas se visualizará en la pantalla o quedará impreso en papel (LPRINT).

• **Ejemplo:** 10PRINT «Esto se visualizará en pantalla»

• **En ausencia de comillas, el ordenador cree que se trata de variables.**

```
5 NEW
10 PRINT «A»
20 PRINT A
RUN
A [presenta el carácter entre comillas]
0 [presenta el valor de la variable A]
```


• Frecuentemente se emplean las comillas **para imprimir espacios en blanco.**

Ejemplo:

```
10 INPUT A$
20 PRINT «Bravo»; « »; A$
```

En ausencia de comillas, se produciría una yuxtaposición entre BRAVO y la variable alfanumérica (por ejemplo ANDRÉS): BRAVOANDRÉS.

• Finalmente, las comillas sirven también para presentar visualmente **el sentido de la pregunta** cuando se utiliza INPUT.

Ejemplo:

```
10 INPUT «Cuál es su nombre»; A$
20 PRINT A$
RUN
¿Cuál es su nombre? Jorge
Jorge
```

• En algunos dialectos pueden hallarse todavía **otras utilizaciones** de las comillas: dar un nombre a un programa, utilizar algunos DATA, etc.

Coma

Este operador, al igual que el anterior, tiene un gran número de usos. Algunos de ellos son «ortodoxos», mientras que otros lo son ya menos.

• Es muy frecuente utilizar la coma **en una instrucción** PRINT para imprimir los resultados a intervalos determinados (desempeña, así, la función de tabulador). La amplitud de estos intervalos depende del ordenador.

• La coma se emplea también **en la instrucción** READ DATA para separar los distintos datos.

• **En la instrucción** INPUT sirve también para separar las variables que hay que entrar en el teclado.

• **Ejemplo**

```
10 INPUT A,B,C,D
20 PRINT A;B;C;D
30 PRINT A,B,C,D
RUN
? 1
? 2
? 3
? 4
1 2 3 4
1    2    3    4
```

Paréntesis

Tienen, fundamentalmente, dos funciones:

• **Funciones matemáticas:** delimitar las operaciones que se efectuarán siguiendo un orden de prioridad.

• En la instrucción DIM, para **dar las dimensiones de una tabla.**

• En cuanto **al orden de las prioridades** en una fórmula matemática, recordemos que es el siguiente:
en prioridad, las operaciones que están entre paréntesis;
después, el orden de las prioridades es el siguiente:
— elevación a la potencia
— negación
— multiplicación
— división
— suma
— resta

Dólar ($)

Sin este símbolo, no habría manera de utilizar cadenas. Colocado después de una letra (A$), **designa una variable alfanumérica;** colocado después de una cadena (GHE$) **designa una variable-cadena.** Todo ello se trata más detalladamente en el capítulo dedicado a las variables.

Digamos una vez más que los dialectos Basic se diferencian bastante entre sí por la manera en que tratan variables:
— a veces, el ordenador solamente tiene en cuenta los dos primeros caracteres (ANDRE$ y ANGUERAN$ se consideran ¡idénticos!);
— la longitud de las cadenas admitidas no es fija;
— es frecuente que esté prohibido utilizar una palabra clave como variable (DIM$ está prohibida).

Sostenido (#)

Este símbolo se emplea para indicar que una variable es del tipo «doble precisión». Ver «Signo de admiración».

Por ciento (%)

Este símbolo se emplea para **indicar que una variable es entera.** Este operador es idéntico a la función INT (ver diccionario).

Hay que advertir que algunos dialectos lo utilizan también para otras indicaciones.

Barra invertida (\)

Idéntica a los **dos puntos.** Más rara

Signo de interrogación

Abreviatura frecuente de la instrucción PRINT.
Ejemplo (en modo directo)

```
? 7*3
21
```

(en vez de PRINT 7*3)

Apóstrofo

Sustituye a REM.

Las constantes y las variables

Las constantes

Una constante es una **magnitud que conserva el mismo valor cada vez que se hace un cálculo o se ejecuta un programa.** Por el contrario, una variable cambia constantemente de valor.

Para dar un ejemplo dentro del campo de las matemáticas, recuérdese que pi es una constante (tiene siempre el mismo valor; 3,1416...), mientras que en una ecuación de segundo grado x, y,... son variables.

En programación, hay constantes numéricas y constantes alfanuméricas.

Las constantes numéricas

Una constante numérica es **un número** (positivo, negativo o nulo) **que contiene un máximo de 8 cifras.** La constante puede contener un punto decimal o un exponente.

- **Ejemplos:**

```
1.234567
8
```


- Las constantes pueden almacenarse de forma entera, en simple precisión (con 7 cifras significativas) o en doble precisión (con 17 cifras significativas, pero con presentación de sólo 16 cifras).

Las constantes alfanuméricas

Una constante alfanumérica es **una cadena de caracteres colocada entre comillas.** La longitud admitida para la cadena depende del ordenador que se utilice.

- **Ejemplos:**

```
«NOMBRE»
«ALBERTO MORA»
```

Las variables

La memoria central del ordenador contiene cierto número de compartimentos en los que se hallan dispuestos unos valores provisionales que sirven en un momento dado y pueden sustituirse rápidamente. **Estos valores provisionales se denominan variables.**

Hay que distinguir, fundamentalmente, tres tipos de variables:

1 — Las variables numéricas

2 — Las variables alfanuméricas

3 — Las variables «dimensionadas». Estudiamos esta última categoría de variables con la instrucción DIM.

Las variables numéricas

Ahora que usted ya conoce bien la instrucción PRINT, efectúe (en modo directo) la instrucción siguiente: PRINT A. El ordenador presenta la visualización Ø. Pruebe lo mismo con B, C, etcétera. Obtendrá siempre Ø (excepto, claro está, que tenga un programa en memoria). Esto significa, sencillamente, que para el ordenador la zona de memoria asignada a la letra A (o, más exactamente, a la variable A) no contiene nada. Esto es completamente normal, ya que no hemos entrado nada. Lo contrario indicaría un mal funcionamiento en la organización interna del ordenador.

Así pues, **una variable es una zona de memoria —identificada por una letra— que contiene datos provisionales.**

• Para las variables numéricas pueden **utilizarse todas las letras que contiene el teclado del ordenador.** Cada letra puede ir seguida o no de una cifra. Ejemplos de variables: A, A9, B6, etc.

• **Estas variables sólo pueden contener cifras.** Conviene advertir que, utilizando estas variables para cifras, se precisará, a veces, si tales variables son únicamente *números enteros* (se utilizará generalmente el signo % para indicar el número entero: A%, B7%, etcétera), *en simple precisión* (lo cual se indica con el signo !: A!, B7!, etcétera). Estos signos de identificación de tipos no son necesariamente los mismos en todos los microordenadores.

• Ejemplo de programa con utilización de variables:

```
10 PRINT «Veremos de qué se trata...»
20 LET A = 3
30 LET B = 4
40 LET C = A+B
50 PRINT C
```

Por tanto, se ha fijado para la variable A el valor 3; para B, el valor 4, y se ha decidido también que C vale A + B. Si se imprime C, se obtendrá el valor 7. A partir de este pequeño programa es posible imaginar muchas otras aplicaciones por el estilo de A-B, A multiplicado por B, etc.

• Las variables numéricas nos servirán durante nuestra utilización del Basic. Es casi imposible escribir un programa sin emplear variables. Por consiguiente, *es indispensable respetar algunas reglas* sencillas pero eficaces:

1 — Confeccionar siempre una tabla de las variables empleadas. En efecto, si sobre la marcha se asigna un nuevo valor a una variable ya empleada, *ésta pierde su antiguo valor y toma el valor nuevo.*

2 — Tener en cuenta que es la parte situada a la izquierda del signo de asignación (=) la que toma el valor situado a la derecha. Así, en LET A = B, A toma el valor de B y no al revés (¡mucho cuidado!).

Las variables alfanuméricas

Como su nombre indica, son variables que pueden contener no solamente cifras, sino también letras. Se las llama también **variables cadenas.**

Una variable que deba contener cadenas de caracteres se reconoce por la presencia del signo dólar ($), que se coloca en última posición:

Así:

A es una variable númerica (sólo puede contener cifras).

A$ es una variable alfanumérica (puede contener cifras y también letras).

• Una cadena de caracteres, para ser válida, ha de estar entre comillas.

• *Ejemplo:*

```
10 LET A$ = «JUVENTUD»
```

La variable A$ es una variable alfanumérica que contiene la cadena de caracteres que forma la palabra JUVENTUD.

• **Adviértase que la cadena de caracteres puede asimismo contener espacios.** Estos espacios son importantes si se desea enlazar dos cadenas en la presentación visual, una al lado de otra. En este caso se prevé un espacio antes o después del último carácter de la cadena. Ejemplo: «JUVENTUD ».

• **Una cadena es limitada en longitud.** El límite suele ser 256 caracteres.

• Cuando se emplean variables alfanuméricas **es indispensable observar las características de su Basic.**

• Algunos ordenadores aceptan variables largas, pero sólo reconocen sus dos primeros caracteres. Así, ÁNGELES y ANTONIETA se considerarán en el programa como una sola y única variable... con los consiguientes errores en la ejecución del programa.

• Una última restricción: **el nombre de una variable no ha de ser una palabra del lenguaje Basic y entre las comillas no se puede utilizar el signo de comillas propiamente dicho,** ya que esto induciría un error en el ordenador y provocaría la aparición en pantalla de un signo de error.

Pequeño programa de utilización de las variables

```
10 PRINT «¿Cuál es su nombre?»
20 INPUT NOMBRE$
30 PRINT «¿Cuál es su edad?»
40 INPUT A
50 PRINT «Su nombre es    »
55 PRINT NOMBRE$
60 PRINT «y su edad es    »
70 PRINT A.
```

Ya se verán otras aplicaciones de las variables cuando se estudien las instrucciones GOTO e IF THEN.

El diagrama de flujo

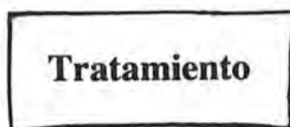
Un diagrama de flujo es la **representación gráfica del desarrollo de las operaciones.** El diagrama de flujo es una etapa importante de la programación que, desafortunadamente, suele ser olvidada por los programadores principiantes. Un buen diagrama, sobre todo si acompaña al programa que usted confeccione, puede ayudarle a encontrar las etapas recorridas y, a la vez, facilitar la lectura del programa a otras personas. El diagrama de flujo, pues, forma parte de la documentación que acompaña a su programa, pero es también una etapa importante en su realización.

En informática se utilizan varios símbolos gráficos para la presentación de los diagramas de flujo; es importante conocer los principales:

• **Principio de un programa:** se utiliza un círculo como éste:



• **Tratamiento de las informaciones en la memoria del ordenador:** se emplea un rectángulo:



■ **Introducción de datos en la memoria:** se utiliza el diseño de la tarjeta perforada (primer soporte para la información):



1

• **Elección en el desarrollo del programa:**

Si hay que elegir entre dos soluciones se utiliza *el rombo:*



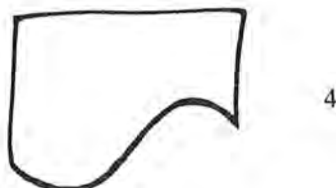
2

Si hay que elegir entre varias soluciones se utiliza *el semicírculo:*



3

• **La impresión de las operaciones** se indica mediante el dibujo de una hoja de papel:



4

• Fin del programa: se emplea el mismo símbolo que para el principio (círculo).

• **Todas estas operaciones están unidas por flechas.**

En el comercio pueden hallarse figuras rígidas recortadas que constituyen una buena ayuda para dibujar estas figuras.

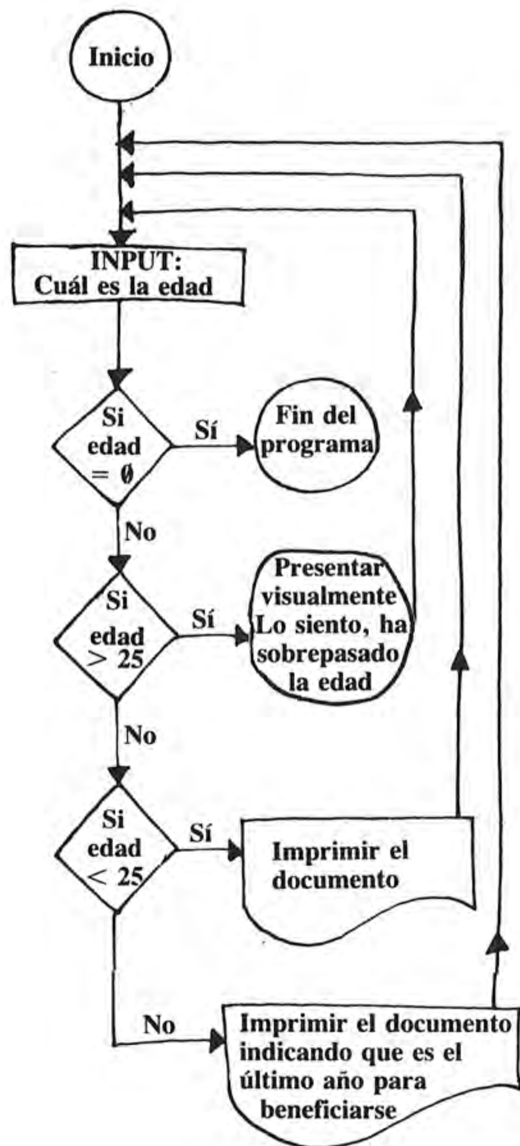
Ejemplo de diagrama de flujo

Se trata de entregar a los alumnos de un centro docente un documento de escolaridad válido para la utilización de transportes públicos. Este documento sólo se libra a los estudiantes cuya edad es igual o inferior a 25 años. Suponiendo que tan sólo van a presentarse los estudiantes normalmente inscritos, lo esencial del programa consiste en no entregar el documento más que a aquellos de edad inferior o igual a 25 años.

El comando RUN únicamente ha de utilizarse una vez y el programa continúa haciendo preguntas. Para detenerlo, basta con pulsar el cero.

Antes de observar el diagrama de flujo dibujado en la página siguiente, pruebe de imaginar las etapas y su representación.

• **Digamos, de paso, que la «receta» para llegar al resultado deseado se denomina algoritmo.** Un algoritmo, a pesar de su aspecto tan digno y su nombre erudito, no es, por tanto, más que la **receta detallada de las operaciones necesarias para obtener un resultado.** En realidad, toda ama de casa elabora algoritmos culinarios a lo largo del año.



- **El programa**

```
1 REM: programa para comprobar la edad
5 PRINT: «Pulse Ø para detener el programa»
1Ø INPUT «¿Cuál es su edad?»; A
15 IF A = Ø GOTO 4ØØ
2Ø IF A > 25 GOTO 3ØØ
25 IF A < 25 GOTO 3Ø
26 LPRINT «Es el último año para beneficiarse de las
reducciones. Este documento le servirá, etc...»
27 GOTO 5
3Ø LPRINT «Este documento le servirá, etc...»
35 GOTO 5
3ØØ PRINT «Lástima, ha sobrepasado la edad requerida»
31Ø GOTO 5
4ØØ PRINT «Programa terminado. Hasta pronto»
45 END
```

- **Comentarios**

— línea 1: línea documental;
— línea 5 y 1Ø: presentación en pantalla;
— líneas 15, 2Ø, 25: si A no es ni igual a cero ni superior o inferior a 25 es que es igual a 25, y el programa continúa;
— líneas 26 y 3Ø: impresas en papel;
— líneas 3ØØ y 4ØØ: presentación en pantalla.

Últimos consejos

Reglas de programación

Un buen programador no se contenta con escribir un programa que funcione, sino que, a la vez, desea que sea fácil de leer, de comprender y que otro usuario que no sea él tenga la posibilidad de modificarlo.

Además, también a él le gustará modificar la versión de base a medida que sus necesidades o competencia así lo exijan. Por tanto, es necesario que el programa esté documentado, fechado y «bautizado». Generalmente, un programa recibe dos **nombres de bautismo:** el primero es el suyo usual (por ejemplo: BIORRITMOS); el segundo se refiere al estado de la versión. La primera versión suele denominarse 1.0. Si aparecen modificaciones menores, la versión se bautizará como 1.1, 1.2, etc. Cuando las modificaciones son más importantes, se cambia la primera cifra y el programa se denomina 2.0 (y luego 2.1, etc.). Así, una nueva versión para el programa BIORRITMO se llamará 2.0.

Por consiguiente, un buen programador ha de respetar unas reglas que podemos enumerar así:

- **Bautizar y fechar** (no olvidar la etiqueta en el soporte magnético).

- **Documentar** (es decir, utilizar la instrucción REM para introducir comentarios en diversos lugares).

- **Estructurar** (o sea, crear bloques de programas y separarlos bien).

- **Elegir las variables** (es decir, utilizar variables tan mnemónicas como sea posible).

- **Confeccionar una lista de las variables** (para trabajar con mayor comodidad y no utilizar dos veces la misma variable).

- **Evitar los GOTO** intempestivos (que hacen el programa ilegible).

- **Resumir** las funciones del programa y anotar las variables en una hoja anexa al programa.

- **Hacer una lista** (si es posible) sobre papel del programa.

Para aumentar la velocidad de ejecución y condensar el programa...

Quizá su microordenador dispone de una memoria pequeña; en este caso es indispensable condensar el programa. Puede conseguirlo de la siguiente manera:
— suprima todas las observaciones;
— suprima las líneas inútiles;
— utilice variables enteras;
— agrupe varias instrucciones en una sola línea.

Esta pequeña «limpieza» aumentará también la velocidad de ejecución de su programa, velocidad que todavía podrá ser mayor utilizando, en las operaciones repetitivas, más las variables que las constantes.

Cómo corregir un programa

Cualquiera que sea el nivel del programador, es casi imposible escribir un programa completo sin cometer errores. Afortunadamente, el Basic, lenguaje interactivo, indica inmediatamente los errores cometidos por los mensajes que aparecen en la pantalla.

El contenido de los mensajes varía en función de los Basic, pero su estructura general es bastante similar.

- **Ejemplo:**

```
10 PRINT; «BUENOS DÍAS»
SYNTAX ERROR
```

El ordenador señala un error de sintaxis (en efecto, hay un «;» inútil). Los mensajes de errores son numerosos y le aconsejamos estudiar con toda atención el manual que acompaña a su ordenador.

• **Para corregir este error,** puede procederse de varias maneras distintas:
— volver a teclear la línea (la nueva línea borra la anterior);
— borrar la línea por medio del comando DELETE y luego escribirla de nuevo;
— pulse el comando EDIT 1Ø que, en la mayoría de los sistemas, dispone el ordenador en modo edición y permite una cómoda corrección del error.

• **Si el error hubiese aparecido antes de que la línea hubiera sido validada** (por el comando RETURN o ENTER), la corrección podría efectuarse de manera más sencilla: volviendo atrás por medio de las teclas de flechas.

Por tanto, es evidente que existen varios medios para corregir un error, lo cual resulta muy útil si se tiene en cuenta que, como ya hemos dicho, de un programa que se entre en el teclado y funcione ya a la primera vez ¡ningún programador guarda memoria!

• Finalmente, es necesario decir todavía que **un programa que funcione no prueba en absoluto que sea correcto:** el error, quizá, sólo aparecerá si se dan ciertas condiciones muy específicas. La corrección de estos errores forma parte también del trabajo del programador: es lo que se denomina **mantenimiento del software.**

Basic y matemáticas

Como ya hemos visto, el Basic es capaz de efectuar las cuatro operaciones fundamentales en modo programación o en modo directo.

Recordemos que la multiplicación se indica mediante el signo *, y la división mediante el signo /.

Además —y en general— también puede elevar al cuadrado un número (lo que se indica con una flecha hacia arriba).

Aparte estas funciones matemáticas elementales, el lenguaje Basic posee asimismo funciones matemáticas intrínsecas, es decir, que forman parte integrante del lenguaje propiamente dicho. Estas funciones dependen del Basic empleado, y tampoco aquí existe estándar. Las principales funciones existentes son las funciones

trigonométricas (coseno, seno y tangente), las logarítmicas, y algunas funciones útiles, como extracción de la raíz cuadrada, determinación del valor absoluto de un número, etc. Otra función interesante es la función RND (RANDOM), que se describe en el diccionario del final de este libro.

Digamos, finalmente, que al usuario le es permitido crear por sí mismo ciertas funciones mediante el empleo de la instrucción DEF (ver diccionario). Por oposición a las funciones que forman parte integrante del Basic (funciones intrínsecas) se hablará, en este caso, de **funciones extrínsecas.**

Como se trata de una introducción al lenguaje Basic y no de un tratado de matemáticas, no nos parece conveniente extendernos más en estas funciones.

PROBLEMAS Y SOLUCIONES EN LENGUAJE BASIC

Ahora que el Basic ya no constituye un secreto para usted, en este capítulo le proponemos una serie de ejercicios en forma de pequeños programas variados que usted mismo redactará, empleando las instrucciones que acaba de aprender.

Para corregirle, hemos incluido las soluciones. Pero vamos a darle un buen consejo: pruebe primero usted solo y, si ya posee un microordenador, compruebe sus programas antes de cualquier corrección.

La máquina le indicará cuáles son las instrucciones erróneas y esto le permitirá la reelaboración de sus programas.

La programación es, ciertamente, una cuestión de lógica, pero también de práctica y de paciencia.

Por tanto, empiece el trabajo y ¡ánimo!

Algunos programas de cálculo

Problema I — La superficie

Escribir el programa que permita calcular la superficie de un triángulo.

Los datos han de introducirse después de haber pulsado la orden RUN.

En la pantalla deberán aparecer los términos siguientes: base= altura=, superficie=.

- **Programa** (solución)

```
10 INPUT "BASE = "; BA
20 INPUT "ALTURA = "; A
30 S = (BA * A)/2
40 PRINT "BASE = ";BA
50 PRINT "ALTURA = ";A
60 PRINT "SUPERFICIE = ";S
70 END
```

- **Ejemplos de ejecución**

```
RUN
BASE = ? 45
ALTURA = ? 12
BASE = ? 45
ALTURA = 12
SUPERFICIE = 270
```



```
RUN
BASE = ? 789
ALTURA = ? 165
BASE = 789
ALTURA = 165
SUPERFICIE = 65092.5
```

- **Observaciones, trucos y consejos**

Este pequeño programa, muy elemental, no requiere ninguna explicación particular.

- Recordemos únicamente —ya que se trata de nuestro primer programa— que la instrucción INPUT puede comportarse como la instrucción PRINT y presentar un mensaje en pantalla. Este mensaje deberá estar entre comillas.
- A pesar de que el mensaje es siempre una interrogación (INPUT pide, en efecto, una respuesta) no es necesario colocar este signo, ya que INPUT lo genera automáticamente.
- *Las respuestas a las preguntas se almacenarán en variables.* En nuestro caso, se trata de variables genéricas que se designarán con el nombre de BA (BASE), A (altura) y S (superficie). Es conveniente designar siempre las variables de manera sugestiva.
- Digamos, para terminar, que el *punto y coma* (;) ha de estar colocado necesariamente detrás del mensaje. Si lo olvida, aparecerá en pantalla un mensaje de error (SYNTAX ERROR).

Problema II — Cálculo del interés

Escribir el programa que permita calcular el interés (I) producido por un capital (C) colocado a un determinado tanto por ciento (R) durante cierto número de meses (tiempo = T).

Se introducen los datos después de haber pulsado la orden RUN.

En la pantalla han de aparecer los términos siguientes: «capitales=», «tiempo=», «tanto por ciento=».

- **Programa** (solución)

```
10 INPUT "CAPITAL = ";C
20 INPUT "TIEMPO = ";T
30 INPUT "TANTO POR CIENTO = ";R
40 I = C*T*(R/1200)
50 PRINT "CAPITAL = ";C
60 PRINT "TIEMPO = ";T
70 PRINT "TANTO POR CIENTO = ";R
80 PRINT "INTERES = ";I
90 END
```

• **Ejemplos de ejecución**

```
CAPITAL 100000
TIEMPO = 12
TANTO POR CIENTO = 10
INTERÉS = 10000

CAPITAL = 452136
TIEMPO = 6
TANTO POR CIENTO = 14.5
INTERÉS 32779.9
```

• **Observaciones, trucos y consejos**

No olvide que una multiplicación se indica, en Basic, mediante el signo *.

Finalmente, el Basic ignora (¡como los anglosajones!) la coma *para indicar los decimales* y utiliza siempre el punto.

Problema III — División selectiva

Escribir el programa que permite introducir dos números en la pantalla y dividir el mayor por el menor únicamente si este último es distinto de Ø.

Utilizará la instrucción IF THEN ELSE.

Los dos números se introducen en la pantalla después de haber pulsado la orden RUN.

• **Programa** (solución)

```
10 INPUT "PRIMER NUMERO: ";A
20 INPUT "SEGUNDO NUMERO: ";B
30 IF A>B THEN 40 ELSE 90
40 IF B=0 THEN 50 ELSE 90
50 PRINT "DIVISION IMPOSIBLE"
80 STOP
90 IF A=0 THEN 100 ELSE 120
100 PRINT "DIVISION IMPOSIBLE"
110 STOP
120 PRINT "RESULTADO = ";A/B
140 END
```

• **Ejemplos de ejecución**

```
RUN
PRIMER NÚMERO: ? 150
```



```
SEGUNDO NÚMERO: ? 3
RESULTADO = 50

RUN
PRIMER NÚMERO: ? 0
SEGUNDO NÚMERO: ? 10
DIVISIÓN IMPOSIBLE

RUN
PRIMER NÚMERO: ? 452
SEGUNDO NÚMERO: ? 26
RESULTADO = 17.3846

RUN
PRIMER NÚMERO: ? 127
SEGUNDO NÚMERO: ? 0
DIVISIÓN IMPOSIBLE
```

Problema IV — Cálculo mental

Se trata de introducir dos números en la pantalla, elegir una operación a efectuar entre las operaciones +, −,* , / y proponer una respuesta.

Si la respuesta propuesta es correcta, imprimir «¡bravo!». Si es errónea, imprimir «respuesta errónea» y volver a pedir otra proposición de respuesta.

No olvide comprobar la operación introducida, es decir, de prever el caso en que se introduce un símbolo distinto de los cuatro símbolos previstos.

• **Programa** (solución)

```
10 INPUT "INTRODUZCA EL PRIMER NUMERO : ";A
20 INPUT "INTRODUZCA EL SEGUNDO NUMERO : ";B
25 PRINT "PRIMER NUMERO = ";A
26 PRINT "SEGUNDO NUMERO = ";B
30 PRINT "ELIJA UNA OPERACION ENTRE : +,-,*,/ "
40 INPUT "SU ELECCION : ";S$
45 PRINT "SU ELECCION : ";S$
50 IF S$ = "+" THEN 110
60 IF S$ = "-" THEN 130
70 IF S$ = "*" THEN 150
80 IF S$ = "/" THEN 170
90 PRINT "SIGNO INCORRECTO"
100 GOTO 40
110 LET R1 = A+B
120 GOTO 220
130 LET R1 = A-B
```

```
140 GOTO 220
150 LET R1 =A*B
160 GOTO 220
170 IF B<>0 THEN 180 ELSE 200
180 LET R1 = A/B
190 GOTO 220
200 PRINT "SI DIVISION, B DEBE SER DIFERENTE DE 0"
210 GOTO 10
220 INPUT  "SU RESPUESTA = ";R
225 PRINT "SU RESPUESTA : ";R
230 IF R1 = R THEN 240 ELSE 260
240 PRINT "BRAVO !"
250 STOP
260 PRINT "RESPUESTA ERRONEA"
270 GOTO 220
280 END
```

- **Ejemplos de ejecución**

```
RUN
PRIMER NÚMERO = ? 100
SEGUNDO NUMERO = ? 52
ELIJA UNA OPERACIÓN ENTRE: +, - *, /
SU ELECCIÓN: ? +
SU RESPUESTA: ? 152
¡BRAVO!

RUN
PRIMER NÚMERO = ? 452
SEGUNDO NÚMERO = ? 52
ELIJA UNA OPERACIÓN ENTRE: =, -, *, /
SU ELECCIÓN: ? -
SU RESPUESTA: ? 403
RESPUESTA FALSA
SU RESPUESTA: ? 400
¡BRAVO!

RUN
PRIMER NÚMERO = 150
SEGUNDO NÚMERO = ? 3
ELIJA UNA OPERACIÓN ENTRE: +, -, *, /
SU ELECCIÓN: ? *
SU RESPUESTA: ? 450
¡BRAVO!

RUN
PRIMER NÚMERO = ? 1000
SEGUNDO NÚMERO = ? 0
ELIJA UNA OPERACIÓN ENTRE: +, -, *, /
SU ELECCIÓN: ? /
```



```
SI DIVISIÓN, B HA DE SER DISTINTO DE 0
PRIMER NÚMERO = ? 1000
SEGUNDO NÚMERO = ? 2
ELIJA UNA OPERACIÓN ENTRE: +, -, *, /
SU ELECCIÓN: ? 500
SIGNO INCORRECTO
SU ELECCIÓN: ? /
SU RESPUESTA: ? 500
¡BRAVO!

RUN
PRIMER NÚMERO = ? 12.5
SEGUNDO NÚMERO = ? 23.6
ELIJA UNA OPERACIÓN ENTRE: +, -, *, /
SU ELECCIÓN: ? =
SIGNO INCORRECTO
SU ELECCIÓN: ? *
SU RESPUESTA: ? 294
RESPUESTA FALSA
SU RESPUESTA ? 295
¡BRAVO!

RUN
PRIMER NÚMERO = ? 156.23
SEGUNDO NÚMERO = ? 45.789
ELIJA UNA OPERACIÓN ENTRE: +, -, *, /
SU ELECCIÓN: ? -
SU RESPUESTA: ? 110.441
RESPUESTA FALSA
SU RESPUESTA: ? 110
RESPUESTA FALSA
```

- **Observaciones, trucos y consejos**

Recuerde que, en Basic, se distinguen *dos clases de variables:*

— las variables numéricas (que son siempre y exclusivamente cifras);

— las variables alfanuméricas (es decir, cualquier signo o símbolo utilizado en lenguaje Basic).

El ordenador reconoce las *variables alfanuméricas* porque van seguidas del símbolo dólar ($). Los *símbolos de operaciones* (+, *, - y /) no son cifras; por consiguiente, son variables alfanuméricas y para designarlas se utilizará el signo dólar ($).

Este programa nos permite también utilizar *la instrucción IF y el salto condicional* GOTO.

En el programa anterior hemos utilizado el símbolo *mayor que* (>); utilizaremos aquí el símbolo *diferente de* (<>).

Problema V — Test

En este programa se le pide que introduzca en la pantalla un número N, compruebe su valor y le aplique una operación aritmética distinta, según sea este valor.
— Si N = 100, es necesario multiplicar por 3.
— Si N > 100, es necesario multiplicar por 2.
— Si N < 100, es necesario elevarlo al cuadrado.
Evidentemente, los resultados han de aparecer en pantalla.

• **Programa** (solución)

```
10 INPUT "NUMERO DE PARTIDA";N
30 IF N=100 THEN 60
40 IF N>100 THEN 90
50 IF N<100 THEN 120
60 N=N*3
70 PRINT N
80 END
90 N=N*2
100 PRINT N
110 END
120 N=N^2
130 PRINT N
140 END
```

• **Ejemplos de ejecución**

```
RUN
NÚMERO DE PARTIDA: ? 45
2025

RUN
NÚMERO DE PARTIDA: ? 125
250

RUN
NÚMERO DE PARTIDA: ? 100
300
```

Problema VI — Nota de gastos

Haga el programa de cálculo e impresión que permita obtener la presentación visual siguiente


```
Nota de gastos

Restaurante        2000
Garaje             1000
Taxi               1200
                   ----
Total              4200
                   ====
```

• **Programa** (solución)

```
10 A=2000
20 B=1000
30 C=1200
40 D=A+B+C
50 PRINT "NOTA DE GASTOS"
60 PRINT "--------------"
70 PRINT "RESTAURANTE",A
80 PRINT "GARAJE",B
90 PRINT "TAXI",C
100 PRINT,"-------"
110 PRINT "TOTAL",D
120 PRINT,"======="
130 END
```

Algunos programas de lógica

Problema I — Vida de estudiante

Programa el precio de venta (PV) de un periódico estudiantil ofrecido a los automovilistas detenidos ante el semáforo en rojo.

Según conduzcan un automóvil pequeño (código 1), uno mediano (código 2) o uno grande (código 3), el precio del periódico será de 50, 100 ó 250 pesetas.

En la pantalla han de aparecer los distintos códigos posibles y le corresponde a usted introducir el código correspondiente al tipo de vehículo detenido ante el semáforo en rojo.

A continuación, siempre en la pantalla, han de aparecer el precio y la observación siguiente: «¡... pesetas por este periódico excepcional! ¡Gracias!»

- **Ejemplos de ejecuciones**

```
RUN
CÓDIGO DE LOS AUTOMÓVILES:
PEQUEÑO = 1
MEDIANO = 2
GRANDE = 3
TIPO DE AUTOMÓVIL: ? 1
¡50 PESETAS POR ESTE PERIÓDICO EXCEPCIONAL!
¡GRACIAS

RUN
TIPO DE AUTOMÓVIL: ? 2
```



```
¡100 PESETAS POR ESTE PERIÓDICO EXCEPCIONAL!
¡GRACIAS!

RUN
TIPO DE AUTOMÓVIL: ? 3
¡250 PESETAS POR ESTE PERIÓDICO EXCEPCIONAL!
¡GRACIAS!

RUN
TIPO DE AUTOMÓVIL: ?0
```

Para interrumpir la ejecución del programa, basta introducir 0 para el tipo de automóvil.

- **Programa** (solución)

```
10 PRINT "CODIGO DE LOS COCHES : "
20 PRINT "PEQUENO = 1 "
30 PRINT "MEDIANO = 2"
40 PRINT "GRANDE = 3"
50 INPUT "TIPO DE COCHE";A
55 PRINT "TIPO DE COCHE : ";A
60 IF A=1 THEN 110
70 IF A=2 THEN 130
80 IF A=3 THEN 150
85 IF A=0 THEN 210
90 PRINT "ERROR DE CODIGO"
100 GOTO 50
110 PRINT "50 ";
120 GOTO 160
130 PRINT "100 ";
140 GOTO 160
150 PRINT "250 ";
160 PRINT "PESETAS POR ESTE PERIODICO EXCEPCIONAL !!"
170 PRINT "...GRACIAS !"
180 PRINT
190 PRINT
200 GOTO 50
210 END
```

- **Observaciones, trucos y consejos**

Este programa le inicia en la noción de MENÚ (o REPERTORIO) y en la de control de retorno.

Un MENÚ es la presentación visual de diversas opciones del programa. En el presente caso, se indica que los automóviles pueden ser pequeños, medianos o grandes. A la pregunta (¿qué tipo de automóvil?) no es preciso responder con todas las letras, sino únicamente pulsando una cifra. Los programas bien estructurados y concebidos para una cómoda utilización permiten visualizar muchos menús.

El control de retorno impide que el programa se «clave» en caso de error en la respuesta. Así, si usted responde algo que no sea uno de los códigos propuestos, el ordenador le señalará el error y —gracias a la instrucción GOTO— esto le permitirá comenzar de nuevo.

Si usted pulsa 0, el programa se detiene. La mayoría de los programadores utilizan esta convención: si se pulsa cero en un menú, el programa se detiene (o vuelve al MENÚ anterior).

Recordemos asimismo que la instrucción PRINT, sola en una línea (líneas 180 y 190), ordena solamente un retorno de carro, dejando así una línea virgen en la pantalla. Se trata, pues, de *una utilización de* PRINT *para la compaginación.*

Problema II — Juego de adivinanza

Escriba un programa que, interrogando al ordenador, permita hallar un número de 0 a 9 introducido previamente en memoria. Deberá responder «exacto» si la respuesta es correcta, «demasiado pequeño» o «demasiado grande» si la respuesta es incorrecta.

El número se introduce en el programa antes de la ejecución (instrucciones READ DATA).

La presentación visual será, por ejemplo, de este tipo:

Número propuesto (de 0 a 9)?
El número... es demasiado pequeño. Introduzca otro
El número... es demasiado grande. Introduzca otro
¡Bravo!, el número es...

- **Ejemplo de ejecución**

```
RUN
NÚMERO PROPUESTO (DE 0 A 9): ? 3
EL NÚMERO 3 ES DEMASIADO PEQUEÑO... INTRODUZCA OTRO
NÚMERO PROPUESTO (DE 0 A 9): ? 5
EL NÚMERO 5 ES DEMASIADO GRANDE... INTRODUZCA OTRO
NÚMERO PROPUESTO (DE 0 a 9): ? 4
¡BRAVO!, EL NUMERO ES: 4
```

- **Programa** (solución)

```
10 READ A
20 INPUT "NUMERO PROPUESTO (DE 0 A 9) : ";X
30 IF X<A THEN 70
40 IF X>A THEN 90
50 PRINT "BRAVO....,EL NUMERO ES : ";A
60 END
70 PRINT "EL NUMERO ";X;"ES DEMASIADO PEQUENO,...
   INTRODUZCA OTRO"
80 GOTO 20
90 PRINT "EL NUMERO ";X;"ES DEMASIADO GRANDE,...
   INTRODUZCA OTRO"
100 GOTO 20
110 DATA 4
```

- **Observaciones, trucos y consejos**

Este pequeño programa, muy sencillo, utiliza la instrucción READ DATA.

Esta instrucción, de la que haremos un gran uso, requiere cierto rigor. Por tanto, será conveniente recordar determinadas reglas:

1 — READ *ha de preceder siempre* a DATA.

2 — READ *puede «leer» datos numéricos o alfanuméricos* (¡no equivocarse al utilizar las variables!)

3 — La naturaleza y el número de las variables empleadas impone el número y la naturaleza de los DATA. En otras palabras, cada READ *ha de tener su* DATA.

4 — Los datos (DATA) se separan mediante una coma. Hay que colocar los datos en una o varias líneas sin que esto cambie nada en la ejecución del programa. Pero ¡atención!: el último dato de una línea no ha de ir seguido de coma.

Problema III — ¡Oh, París!

Utilizando tan sólo 3 variables alfanuméricas, escriba el programa que permita seleccionar un país y su capital, introduciendo el nombre del país.

Los cinco países y su capital se colocan en DATA.

Tendrá, por ejemplo, la siguiente visualización:

El programa incluye 5 países y su capital
Introduzca el nombre del país y obtendrá su capital
? Francia
La capital de Francia es París

- **Programa** (solución)

```
10 PRINT "EL PROGRAMA INCLUYE 5 PAISES Y SU CAPITAL"
20 PRINT "INTRODUZCA EL NOMBRE DEL PAIS ......
25 PRINT ".... Y OBTENDRA SU CAPITAL"
30 INPUT P$
```

```
40 FOR A=1 TO 5
50 READ A$,B$
60 IF P$=A$ THEN 150
70 NEXT A
80 PRINT
90 PRINT
100 PRINT "ESTE PAIS NO ESTA EN EL PROGRAMA"
120 PRINT "VUELVA A EMPEZAR"
125 RESTORE
130 GOTO 10
140 PRINT
150 PRINT P$;" SU CAPITAL ES ";B$
160 DATA BELGICA,BRUSELAS,REINO UNIDO,LONDRES
170 DATA SUECIA,ESTOCOLMO,FRANCIA,PARIS,SUIZA,BERNA
190 END
```

- **Ejemplos de ejecución**

```
RUN
EL PROGRAMA INCLUYE 5 PAISES Y SUS CAPITALES INTRO-
DUZCA EL NOMBRE DEL PAÍS Y OBTENDRÁ SU CAPITAL
? FRANCIA
LA CAPITAL DE FRANCIA ES PARÍS

RUN
EL PROGRAMA INCLUYE 5 PAÍSES Y SUS CAPITALES INTRO-
DUZCA EL NOMBRE DEL PAÍS Y OBTENDRÁ SU CAPITAL
? REINO UNIDO
LA CAPITAL DEL REINO UNIDO ES LONDRES

RUN
EL PROGRAMA INCLUYE 5 PAÍSES Y SUS CAPITALES INTRO-
DUZCA EL NOMBRE DEL PAÍS Y OBTENDRÁ SU CAPITAL
? AUSTRIA
ESTE PAÍS NO ESTÁ EN EL PROGRAMA
VUELVA A EMPEZAR
```

- **Observaciones, trucos y consejos**

Este pequeño programa utiliza también la instrucción READ DATA.

Los datos son alfanuméricos; por tanto, también lo son las variables.

Se introducen en DATA cinco países y sus capitales. *Los datos se separan por comas:* PAÍS, CAPITAL, PAÍS, CAPITAL, etcétera.

En la línea 40, *un bucle obliga al programa a leer todos los datos* (en número de cinco).

Una variable se utiliza para los países (A$) y otra para las capitales (B$). En la instrucción READ, las dos variables alfanuméricas están separadas por una coma; por tanto, el ordenador «sabe» que ha de leer *los datos de dos en dos.*

Como nuestro programa sólo conoce cinco países y el usuario ignora cuáles, si el país no figura en datos (es lo que el ordenador comprueba en la línea 60), el ordenador indica que se vuelva a empezar.

El ordenador recomienza a cero la lectura de los datos (esto se obtiene con la instrucción RESTORE).

En la línea 150 es *indispensable prever unos blancos* antes de la presentación visual de las variables; de lo contrario, en la pantalla se visualizará, por ejemplo: LACAPITALDEFRANCIAESPARÍS, lo cual es cierto, pero antiestético.

Si usted ha comprendido bien la estructura de este programa, podrá ya empezar a escribir muchos pequeños programas didácticos o divertidos (la edad del tío Ramón o el número de dientes del bebé).

Problema IV — ¿Quién es quién?

Escriba un programa que permita seleccionar un nombre y una dirección en una lista, dando solamente el nombre. Prevea la presentación visual de «nombre desconocido» para el caso de que se introduzca un nombre que no figure en la lista.

- **Programa** (solución)

```
20 INPUT "NOMBRE";C$
25 FOR A=1 TO 3
30 READ A$,B$
35 IF C$=A$ THEN 80
45 NEXT A
60 PRINT "NOMBRE DESCONOCIDO"
70 END
80 PRINT A$,B$
90 DATA ROMAN, AVDA.BRASIL 6 28020 MADRID
100 DATA SALA, NUMANCIA 91 08029 BARCELONA
110 DATA GUZMAN, TURIA 5 41011 SEVILLA
```

- **Ejemplos de ejecución**

```
RUN
¿NOMBRE? ROMÁN
ROMÁN                    AVDA BRASIL 6 28020 MADRID
```

```
RUN
¿NOMBRE? SALA
SALA                    NUMANCIA 91 08029 BARCELONA

RUN
¿NOMBRE? GUZMAR
NOMBRE DESCONOCIDO

RUN
¿NOMBRE? GUZMÁN
GUZMÁN                  TURIA 5 41011 SEVILLA
```

- **Observaciones, trucos y consejos**

Este programa se parece mucho al anterior. Obsérvese que cada línea de DATA comprende un dato completo.

Advirtamos que *no hay que colocar una coma al final de una línea* de DATA.

A partir de este programa *—y modificando la línea 25* en función de los datos contenidos en el programa— podrá formar un pequeño fichero personal (listín telefónico, carné de direcciones, lista de libros, etc...)

El programa siguiente tiene la misma estructura de base, pero está adaptado a un programa bibliográfico.

Cada vez que entre en posesión de un nuevo libro *basta modificar la línea 25 y entrar las características del libro en DATA.*

Problemas IV bis — Bibliografía

Escriba un programa que le dé informaciones sobre los autores de su biblioteca (nombre de pila, obras en la biblioteca, editor, año de edición) dando, simplemente, el apellido del autor. Prevea el caso de no tener ninguna obra de dicho autor.

- **Programa** (solución)

```
10 REM BIBLIOGRAFIA
20 INPUT "NOMBRE DEL AUTOR";F$
25 FOR A= 1 TO 10
30 READ A$,B$,C$,D$,E$
35 IF F$=A$ THEN 80
45 NEXT A
60 PRINT "AUTOR DESCONOCIDO"
65 RESTORE
66 GOTO 190
70 END
80 PRINT A$,B$,C$,D$,E$
90 DATA FREUD,SIGMUND,INTERPRETACION DE LOS SUENOS,
   ALIANZA,1981
100 DATA BALZAC,HONORATO,PAPA GORIOT,JUVENTUD,1980
110 DATA HUGO,VICTOR,POESIAS,AGUILAR,1983
120 DATA CERVANTES,MIGUEL DE,ENTREMESES,JUVENTUD,1984
130 DATA DOSTOIEWSKI,FEDOR,CRIMEN Y CASTIGO,JUVENTUD,1985
140 DATA GOETHE,JUAN W.,FAUSTO,ESPASA CALPE,1981
150 DATA BALMES,JAIME,EL CRITERIO,JUVENTUD,1979
160 DATA SANTA TERESA,-,LAS MORADAS,JUVENTUD,1982
170 DATA GIDE,ANDRES,EL INMORALISTA,ARGOS VERGARA,1981
180 DATA TOLSTOI,LEON,ANA KARENINA,JUVENTUD,1983
190 INPUT "DESEA OTRO TITULO? (SI/NO)";X$
200 IF X$= "NO" THEN GOTO 1000
210 RESTORE
220 GOTO 10
1000 END
```

Problemas V — Las fechas

Escriba el programa de control de verosimilitud de la introducción de una fecha en INPUT (2 cifras para los días, dos cifras para los meses y cuatro cifras para los años —que han de estar comprendidos entre 1900 y 2000).

Pida, a continuación, la impresión de la fecha.

- **Programa** (solución)

```
10 INPUT"DIA (DD)";D
30 IF D<1 THEN 10
40 IF D>31 THEN 10
50 INPUT"MES (MM)";M
70 IF M<1 THEN 50
80 IF M>12 THEN 50
90 INPUT"ANO (AAAA)";A
110 IF A<1900 THEN 90
120 IF A>2000 THEN 90
130 PRINT D;"/";M;"/";A
140 END
```

- **Ejemplos de ejecución**

```
RUN
DÍA (DD) ? 25
MES (MM) ? 06
AÑO (AAAA) ? 1975
25/06/1975

RUN
DÍA (DD)? 0
```

```
DÍA (DD) ? 2
MES (MM) ? 14
MES (MM) ? 11
AÑO (AAAA) ? 1983
2/11/1983

RUN
DÍA (DD) ? 31
MES (MM) ? Ø7
AÑO (AAAA) ? 1895
AÑO (AAAA) ? 1956
31/Ø7/1956
```

- **Observaciones, trucos y consejos**

Este pequeño programa contiene un *test* de verosimilitud para los días del mes: si el número de días es inferior a 1 o superior a 31, la introducción es rechazada.

Como ejercicio, le proponemos completar este programa añadiéndole las instrucciones necesarias para controlar el mes de febrero (por ejemplo, utilizando IF THEN ELSE) y luego —¿por qué no?— todos los meses del año.

Problema VI — Agencia de viajes

Escriba el programa que permita seleccionar en un fichero los hombres de más de 40 años, solteros y con unos ingresos de más de 150.000 pesetas al mes.

Se pide la visualización de las informaciones siguientes para cada persona seleccionada:

N.º de matrícula   Edad   Ingresos

Introduzca los datos en DATA (para 10 personas, por ejemplo). Introduzca observaciones (instrucción REM) en su programa para indicar que:

```
N = Número
B = Sexo  B1 = Hombre
          B2 = Mujer
C = Edad
D = Estado civil  D1 = Soltero
                  D2 = Casado
E = Ingresos
```


- **Programa** (solución)

```
10 REM : N=NUMERO
20 REM : B=1=HOMBRE,B=2=MUJER
30 REM : C=EDAD
40 REM : D=1=SOLTERO,D=2=CASADO
50 PRINT "NUMERO","NOMBRE","EDAD","INGRESOS"
60 REM : E=INGRESOS
70 READ N,A$,B,C,D,E
80 IF N=-2 THEN END
90 IF B=2 THEN 70
100 IF C<=40 THEN 70
110 IF D=2 THEN 70
120 IF E<=150000 THEN 70
130 PRINT N,A$,C,E
140 GOTO 70
150 DATA 145,SOLER,1,45,2,80000
160 DATA 185,MARTIN,1,50,1,180000
170 DATA 174,RUIZ,2,25,1,120000
180 DATA 210,PARES,1,41,2,160000
190 DATA 214,SOTILLO,2,36,2,140000
200 DATA 241,MORO,1,48,2,165000
210 DATA 251,SANTOS,2,20,1,130000
220 DATA 256,APARICIO,1,51,2,170000
230 DATA 264,VERDU,1,29,2,190000
240 DATA 273,GIL,1,58,1,175000
250 DATA -2,Z,0,0,0,0
```

- **Ejecución**

```
RUN
 185        MARTIN      50          180000
 273        GIL         58          175000
```

Si deseamos obtener la lista de las personas casadas que cumplan las mismas condiciones, basta sustituir, en la línea 11Ø D = 2 por D = 1.

```
210        PARES       41          160000
241        MORO        48          165000
256        APARICIO    51          170000
```

- **Observaciones, trucos y consejos**

Este programa de selección está totalmente basado en un fichero contenido en DATA. Ya hemos encontrado esta instrucción en otras ocasiones; por tanto, no es necesario recordar sus puntos esenciales. La selección de informaciones se realiza empleando la instrucción IF THEN para cada condición (sexo, salario y edad).

También hay en este programa una pequeña argucia de pro-

gramación que se denomina **señal indicadora** (en inglés, «flag») y que aparece en la línea 80.

Una señal indicadora permite introducir una derivación.

De momento, limítese a teclear este programa tal cual lo presentamos. Pero si le falta paciencia y desea comprender inmediatamente la función de la señal indicadora, vea la explicación que se da en el programa de gestión n.º 4.

Problemas VII — Calor y frío

Se trata de crear un pequeño programa que transforme una temperatura dada en grados Celsius (Europa) y su equivalente en grados Fahrenheit (EE.UU., por ejemplo); y viceversa.

Para hacer el programa algo más complejo, crear también un pequeño menú que le permita seleccionar la conversión elegida:

1. EE.UU → Europa
2. Europa ← EE.UU.

Las variables utilizadas son todas numéricas:

C: para los grados Celsius
F: para los grados Fahrenheit
N: para la elección en función del menú (o repertorio)
X e Y (por tradición): para los cálculos.

- **Programa** (solución)

```
10 REM CONVERSION FAHRENHEIT/CELSIUS
20 PRINT "ELIJA LA CONVERSION DESEADA:"
30 PRINT" 1 = GRADOS FAHRENHEIT EN GRADOS CELSIUS"
40 PRINT" 2 = GRADOS CELSIUS EN GRADOS FAHRENHEIT"
50 INPUT " 1 O 2";N
60 IF N>2 THEN 2000
61 IF N=2 THEN 1000
65 PRINT "CONVERSION TEMPERATURA FAHRENHEIT EN CELSIUS"
70 INPUT "INDIQUE LA CIFRA A CONVERTIR";F
80 X= (F-32)*5/9
85 PRINT
90 PRINT "LA RESPUESTA ES";X
100 PRINT "GRADOS CELSIUS"
110 END
1000 PRINT "CONVERSION TEMPERATURA CELSIUS EN FAHRENHEIT"
1005 INPUT "INDIQUE LA TEMPERATURA A CONVERTIR";C
1010 Y= C*(9/5)+32
1020 PRINT "LA RESPUESTA ES";Y
1030 PRINT "GRADOS FAHRENHEIT"
1040 END
2000 PRINT "PONGASE LAS GAFAS.....: 1 O 2 !"
2010 GOTO 10
```


- **Pequeño consejo... para la publicación**

Si desea publicar sus programas, es necesario dar no solamente el listado, sino también algunos ejemplos de ejecución (como en este libro).

Para reproducir en papel lo que se visualiza en la pantalla es necesario traducir todos los PRINT en LPRINT. Además, no hay que olvidar escribir de nuevo los INPUT empleando la declaración PRINT (sin omitir el signo de interrogación, que es colocado automáticamente por INPUT, pero no por PRINT o LPRINT).

A título de ilustración damos seguidamente el listado del programa anterior tal como se ha preparado para la reproducción en esta obra:

```
10 REM CONVERSION FAHRENHEIT/CELSIUS
20 LPRINT "ELIJA LA CONVERSION DESEADA:"
30 LPRINT" 1 = GRADOS FAHRENHEIT EN GRADOS CELSIUS"
40 LPRINT" 2 = GRADOS CELSIUS EN GRADOS FAHRENHEIT"
50 LPRINT" 1 O 2 ?"
51 INPUT " 1 O 2";N
52 LPRINT N
60 IF N>2 THEN 2000
61 IF N=2 THEN 1000
65 LPRINT "CONVERSION TEMPERATURA FAHRENHEIT EN CELSIUS"
70 LPRINT "INDIQUE LA CIFRA A CONVERTIR?"
71 INPUT F
72 LPRINT F
80 X= (F-32)*5/9
85 PRINT
90 LPRINT "LA RESPUESTA ES";X
100 LPRINT "GRADOS CELSIUS"
110 END
1000 LPRINT "CONVERSION TEMPERATURA CELSIUS EN FAHRENHEIT"
1005 LPRINT "INDIQUE LA TEMPERATURA A CONVERTIR"
1006 INPUT C
1007 LPRINT C
1010 Y= C*(9/5)+32
1020 LPRINT "LA RESPUESTA ES";Y
1030 LPRINT "GRADOS FAHRENHEIT"
1040 END
2000 LPRINT "PONGASE LAS GAFAS.....: 1 O 2 !"
2010 GOTO 10
```

- **Ejemplos de ejecución**

```
ELIJA LA CONVERSION DESEADA:
1 = GRADOS FAHRENHEIT EN GRADOS CELSIUS
2 = GRADOS CELSIUS EN GRADOS FAHRENHEIT
1 O 2 ?
9
PONGASE LAS GAFAS.....: 1 O 2 !
```

```
ELIJA LA CONVERSION DESEADA:
1 = GRADOS FAHRENHEIT EN GRADOS CELSIUS
2 = GRADOS CELSIUS EN GRADOS FAHRENHEIT
1 O 2 ?
2
CONVERSION TEMPERATURA CELSIUS EN FAHRENHEIT
INDIQUE LA TEMPERATURA A CONVERTIR
1ØØ
LA RESPUESTA ES 212
GRADOS FAHRENHEIT
ELIJA LA CONVERSION DESEADA:
1 = GRADOS FAHRENHEIT EN GRADOS CELSIUS
2 = GRADOS CELSIUS EN GRADOS FAHRENHEIT
1 O 2 ?
1
CONVERSION TEMPERATURA FAHRENHEIT EN CELSIUS
INDIQUE LA CIFRA A CONVERTIR ?
32
LA RESPUESTA ES Ø
GRADOS CELSIUS
```

Nota: Adviértase que el punto de ebullición corresponde a 212° F (siempre es útil saberlo).

Nota: 32° F es el punto de congelación.

Problema VIII — Problema culinario

El pequeño problema que ahora le proponemos tiene un doble interés:

— por una parte, puede usted utilizar totalmente la estructura del programa anterior (empleando el editor de su microordenador sólo tendrá que introducir unos cambios mínimos).

— por otra parte, interesará a la cocinera de la casa... que, a menudo, es el elemento más refractario a la informática (de esta manera, sus retrasos a las comidas le serán perdonados y las recetas inglesas resultarán siempre «deliciosas»).

Se trata de convertir kilos en libras y viceversa. Tendrá que utilizar la estructura anterior, cambiando sólo los títulos y las variables:

P: por pounds (libras)

K: por kilogramos

Si ha olvidado la relación entre los kilogramos y las libras, le recordaremos que 1 libra equivale a 0,4536 kg.

- **Ejemplos de ejecución**

```
ELIJA LA CONVERSION DESEADA:
1 = KILOGRAMOS EN LIBRAS
2= LIBRAS EN KILOGRAMOS
 1 O 2 ?
 7
PONGASE LAS GAFAS.....: 1 O 2 !
ELIJA LA CONVERSION DESEADA:
1 = KILOGRAMOS EN LIBRAS
2= LIBRAS EN KILOGRAMOS
 1 O 2 ?
 1
CONVERSION KILOGRAMOS EN LIBRAS
INDIQUE EL PESO A CONVERTIR
 1
LA RESPUESTA ES 2.2Ø46
LIBRAS
ELIJA LA CONVERSION DESEADA:
1 = KILOGRAMOS EN LIBRAS
2= LIBRAS EN KILOGRAMOS
 1 O 2 ?
 2
CONVERSION LIBRAS EN KILOGRAMOS
INDIQUE EL PESO A CONVERTIR
 2.5
LA RESPUESTA ES 1.134
KILOGRAMOS
```

- **Programa** (solución)

```
1Ø REM CONVERSION KILOGRAMOS/LIBRAS
2Ø PRINT "ELIJA LA CONVERSION DESEADA:"
3Ø PRINT" 1 = KILOGRAMOS EN LIBRAS"
4Ø PRINT" 2 = LIBRAS EN KILOGRAMOS"
5Ø INPUT " 1 O 2";N
6Ø IF N>2 THEN 2ØØØ
61 IF N=2 THEN 1ØØØ
65 PRINT "CONVERSION KILOGRAMOS EN LIBRAS"
7Ø INPUT "INDIQUE EL PESO A CONVERTIR";K
8Ø X=K* 2.2Ø46
85 PRINT
9Ø PRINT "LA RESPUESTA ES";X
1ØØ PRINT "LIBRAS"
11Ø END
1ØØØ PRINT "CONVERSION LIBRAS EN KILOGRAMOS"
1ØØ5 INPUT "INDIQUE EL PESO A CONVERTIR";P
1Ø1Ø Y=P*.4536
1Ø2Ø PRINT "LA RESPUESTA ES";Y
1Ø3Ø PRINT "KILOGRAMOS"
1Ø4Ø END
2ØØØ PRINT "PONGASE LAS GAFAS......: 1 O 2 !"
2Ø1Ø GOTO 1Ø
```

• **Nota:** utilizando el mismo programa, puede usted (mediante adaptaciones menores) cambiar los kilómetros en millas, los litros en galones, los lis (li: unidad de longitud china de valor variable, comprendido entre 575 y 671 metros) o las verstas en kilómetros (esto último es muy útil cuando se leen obras de Tolstoi o Dostoiewski).

Problema IX — Clasificación

Escriba el programa que le permita clasificar por orden decreciente una serie de 12 números tomados de una lista determinada.

Los 12 números están dados en DATA y han de aparecer en pantalla antes de la clasificación.

• **Programa** (solución)

```
5 DIM V(12)
1Ø FOR I=1  TO 12
2Ø READ V(I)
3Ø PRINT V(I);
4Ø NEXT I
5Ø A=Ø
6Ø FOR I=1 TO 11
65 IF V(I)>=V(I+1) THEN 1ØØ
66 T=V(I)
7Ø V(I)=V(I+1)
8Ø V(I+1)=T
9Ø A=1
1ØØ NEXT I
11Ø PRINT
12Ø IF A=1 THEN 5Ø
13Ø FOR I=1 TO 12
14Ø PRINT V(I);
15Ø NEXT I
16Ø DATA 1,4,7,34,-5,345,9Ø,Ø,-9,9Ø,8,6
17Ø END
```

• **Ejecución**

```
RUN
1   4   7   34   -5   345   9Ø   Ø   -9   9Ø   8   6

345   9Ø   9Ø   34   8   7   6   4   1   Ø   -5   -9
```

• **Observaciones, trucos y consejos**

Este pequeño programa, muy corto, recurre ya a unas nociones avanzadas de programación.

La primera línea recurre a *una instrucción* DIM: se crea una tabla que contiene 12 variables [V(12)]. A continuación, se recurre a un bucle FOR NEXT y a la instrucción READ. Esta última leerá los datos en DATA. Para hacerlo, se le ordena que ejecute el bucle de lectura 12 veces (la variable I tomará sucesivamente los valores 1 a 12). Luego, se le pide que presente en pantalla el resultado de su lectura: PRINT V (I).

La segunda parte del programa (líneas 5Ø a 12Ø) sirve para poner orden en las cifras en DATA, comparando una cifra [V(I)] con la siguiente [V(I+1)]. Las líneas 5Ø, 9Ø y 12Ø sólo tienen por objeto obligar al programa a volver a tomar el bucle en tanto no haya terminado el programa.

Las líneas 65 a 1ØØ sirven para comparar los números, de manera que queden colocados en el orden deseado. Las líneas 13Ø a 15Ø sirven para leer las variables colocadas en orden y presentarlas en pantalla.

Problema X — El mayor

Escriba el programa que permita hallar el número mayor en una lista de 10 números.

• **Programa** (solución)

```
1Ø DIM A(1Ø)
2Ø INPUT "MAGNITUD : ";N
3Ø PRINT "LONGITUD DE LA LISTA : ";N
4Ø LET I=Ø
5Ø LET I=I+1
6Ø INPUT A(I)
7Ø PRINT A(I)
8Ø IF I<N THEN 5Ø
9Ø LET I=1
1ØØ LET X=A(I)
11Ø LET I=I+1
12Ø IF X>= A(I) THEN 14Ø
13Ø LET X=A(I)
14Ø IF I<N THEN 11Ø
15Ø PRINT "MAXIMO : ";X
16Ø END
```

• **Ejemplo de ejecución**

```
RUN
LONGITUD DE LA LISTA: ? 1Ø
? 36
? 89
```

```
? 456
? 852
? 32
? 354
? 3
? 0
? 8
? 10
MAXIMO: 852
```

• **Observación:** el programa se interrumpe automáticamente cuando se llega al número de entradas pedido (en este caso, 10).

Algunos programas de gestión

Problema I — Factura

Haga el documento que permita facturar, ya sea un automóvil (ref. 100), precio unitario 1.000.000 ptas., IVA 33 %, ya sea un vehículo comercial (ref. 200), precio unitario 1.500.000 ptas, IVA 12 %.

Durante la ejecución del programa, sólo se introducen la cantidad y la referencia. Los otros datos se introducen en DATA.

Prevea un *test* de verosimilitud para la referencia.

La presentación en pantalla será la siguiente:

```
Cantidad: ? 2

Referencia: ? 200

                         FACTURA

Cantidad      Referencia      Precio unitario      Total
   2             200             1.500.000       3.000.000
                                 IVA 12%           360.000

  TOTAL                                          3.360.000
```

• **Observaciones, trucos y consejos**

Este pequeño programa hace un uso muy generoso de la **instrucción** TAB. Esta instrucción tiene la misma función que el ta-

bulador de una máquina de escribir, pero le da la posibilidad de colocar un texto o un resultado cualesquiera en el lugar deseado.

Ejemplos:

10 PRINT TAB (30) «Estoy contento»

Presentará visualmente:

Estoy contento

10 PRINT TAB (10) «Estoy contento»

presentará visualmente:

Estoy contento.

También en este programa aparece una argucia matemática que consiste en redondear un número añadiéndole 0.5.

En algunos cálculos, los totales pueden presentar una parte decimal. Como es costumbre redondear el importe al valor entero superior o inferior, y la función INT (INT es la abreviatura de INTEGER, que significa número entero —ver diccionario al final del libro—) redondea siempre el valor inferior, se emplea una argucia que consiste en añadir 0.5.

Ejemplos:

−150,2 se redondea a −151
−150,8 se redondea a −151
+150,2 se redondea a +150
+150,8 se redondea a +150

Por lo tanto, la función INT da el entero mayor inferior al número; esto significa que un número positivo se hace menos positivo y que un número negativo se hace menos negativo.

Para obtener el redondeo correcto basta, pues, añadir 0.5 al número y redondear antes de utilizar la función INT.

Ejemplos:

150,8 + 0.5 = 151,3 redondeado: 151
150,2 + 0,5 = 150,7 redondeado: 150

- **Ejemplos de ejecución**

FACTURA

CANTIDAD	REFERENCIA	PRECIO UNITARIO	TOTAL
7	100	1.000.000	7.000.000
		IVA 33%	2.310.000
			9.310.000

FACTURA

CANTIDAD	REFERENCIA	PRECIO UNITARIO	TOTAL
14	200	1.500.000	21.000.000
		IVA 12%	2.520.000
			23.520.000

- **Programa**

```
10 READ A,B,C,D
20 INPUT "CANTIDAD : ";Q
30 INPUT "REFERENCIA : ";R
40 IF R=100 THEN 80
50 IF R=200 THEN 110
60 PRINT "ERROR DE REFERENCIA"
70 GOTO 30
80 P=A*Q
90 IV=INT(P*(B/100)+.5)
100 GOTO 140
110 REM : VEHICULO COMERCIAL
120 P=C*Q
130 IV=INT(P*(D/100)+.5)
140 MT=P+IV
150 REM : PRESENTACION DE LA FACTURA
160 PRINT TAB(30)"FACTURA"
170 PRINT TAB(30)"======="
180 PRINT TAB(0)"Q";TAB(10)"REF.";TAB(21)"P.U.";
    TAB(54)"TOTAL"
190 PRINT Q;TAB(10)R;
200 IF R=200 THEN 240
210 PRINT TAB(21)A;TAB(54)P
220 PRINT TAB(21)"IVA";B;" %";TAB(55)IV
230 GOTO 260
240 PRINT TAB(21)C;TAB(54)P
250 PRINT TAB(21)"IVA";D;" %";TAB(55)IV
260 PRINT TAB(54)"--------"
270 PRINT TAB(54)MT
280 PRINT TAB(54)"======="
290 DATA 1000000,33,1500000,12
```

Problema II — Prima

Se ha concedido una prima de fin de año en función de los elementos siguientes:

— antigüedad: hasta 10 años: 10.000 pesetas
más de 10 años: 15.000 pesetas
— curso de promoción social: + 5.000 pesetas
— 4.000 por hijo a cargo del trabajador

— no se concede ninguna prima en caso de más de 5 ausencias injustificadas durante el año.

Escriba el programa que permita calcular la prima total de 2 empleados y dé esta visualización:

HOJA DE PRIMAS

N. DEL TRABAJADOR: 1	
PRIMA DE BASE	15.000
SOBREPRIMA CURSO DE PROMOCIÓN SOCIAL	5.000
SOBREPRIMA POR HIJOS	12.000
PRIMA TOTAL	32.000

HOJA DE PRIMAS

N. DEL TRABAJADOR: 2
SE SUPRIME EL DERECHO A PRIMAS
DEBIDO A 6 AUSENCIAS INJUSTIFICADAS

- **Observación:** los datos se introducen en DATA.

- **Programa** (solución)

```
10 REM : CALCULO DE LA PRIMA DE FIN DE ANO
20 P1=0
30 P2=0
40 P3=0
50 P4=0
60 READ A
70 IF A=-2 THEN END
80 READ B,C,D,E
90 IF E>5 THEN 180
100 REM : A=NUM,B=ANTIGUEDAD,C=CURSO P.S.,D=HIJOS
110 IF B>10 THEN 140
120 P1=10000
130 GOTO 150
140 P1=15000
150 IF C=1 THEN 160 ELSE 170
160 P2=5000
170 P3=4000*D
180 P4=P1+P2+P3
190 PRINT "HOJA DE PRIMA"
200 PRINT "-------------"
210 PRINT "NUMERO DEL TRABAJADOR : ";A
220 PRINT
230 IF E>5 GOTO 330
240 PRINT "PRIMA DE BASE";TAB(50)P1
250 PRINT "SUPRIMIDO CURSO P.S.";TAB(51)P2
260 PRINT "SUPRIMIDO HIJOS A CARGO";TAB(51)P3
```



```
270 PRINT TAB(50)"-------"
280 PRINT "PRIMA TOTAL";TAB(50)P4
290 PRINT
300 PRINT
320 GOTO 10
330 PRINT "NO TIENE DERECHO A PRIMA"
340 PRINT "DEBIDO A LAS ";E;" AUSENCIAS INJUSTIFICADAS."
350 GOTO 20
360 DATA 1,14,1,3,2
370 DATA 2,8,0,0,6
380 DATA -2
```

- **Observaciones, trucos y consejos**

Este pequeño programa utiliza una argucia de programación que se llama señal indicadora (ver líneas 70 y 380). Para más explicaciones, ver el programa n.° 4 (Hoja de salario).

Problema III — Salarios

Escriba un programa de cálculo de salarios para 5 obreros designados por su nombre de pila, cada uno de los cuales tiene su número de matrícula.

Los datos, es decir, el número de matrícula, el nombre de pila, el número de horas de trabajo y el salario por hora se introducen en DATA.

La visualización comprenderá, pues, 5 columnas: número, obrero, n.° de horas, salario por hora, salario.

- **Programa** (solución)

```
10 REM: PROGRAMA SALARIOS
20 DIM S(5,3),N$(5)
30 REM : TABLA NUMERICA
40 FOR A=1 TO 5
50   FOR B=1 TO 2
60   READ S(A,B)
70   NEXT B
80 NEXT A
90 REM : TABLA ALFANUMERICA
100 FOR C=1 TO 5
110 READ N$(C)
120 NEXT C
130 REM : CALCULO DEL SALARIO
140 FOR A=1 TO 5
150 S(A,3)=S(A,1)*S(A,2)
160 NEXT A
170 REM LA PARTICULAR DISPOSICION DEL PROGRAMA SE DEBE
180 REM A LOS IMPERATIVOS TECNICOS DE COMPAGINACION
190 REM DE ESTE LIBRO
```

```
200 PRINT"NUM";TAB(7)"OPERARIO";TAB(20)"N.DE HORAS";
210 PRINT TAB(40)"SAL.HORA";TAB(56)"SALARIO"
220 PRINT "----------------------------------";
230 PRINT "----------------------------------"
240 FOR I=1 TO 5
250 PRINT I;TAB(7)N$(I);TAB(25)S(I,1);TAB(44)S(I,2);
    TAB(57)S(I,3)
260 PRINT "----------------------------------";
270 PRINT "----------------------------------"
280 NEXT I
290 DATA 39,575,38,792,33,715,40,830,42,843
300 DATA PEDRO,MIGUEL,JUAN,ROBERTO,LUCAS
```

- **Ejecución**

NUM	OPERARIO	N.DE HORAS	SAL.HORA	SALARIO
1	PEDRO	39	575	22425
2	MIGUEL	38	792	30096
3	JUAN	33	715	23595
4	ROBERTO	40	830	33200
5	LUCAS	42	843	35406

- **Observaciones, trucos y consejos**

Este programa contiene una tabla DIM con 5 líneas y 3 columnas en las cuales se hallan los datos. Algunos de estos datos son numéricos y otros alfanuméricos. Ahora bien, durante el tratamiento es imposible mezclar variables de distinta naturaleza.

Líneas 40 a 80 y 230: Los datos utilizados para los cálculos se colocan en las variables numéricas. *La variable A contiene los números matrícula* (visualización en 5 líneas). *La variable B contiene el número de horas y el salario por hora* (presentación en 2 columnas). Las 4 líneas utilizan el bucle FOR TO. Para los números matrícula, el bucle ha de leer 5 números (FOR A = 1 TO 5); para el número de horas y el salario, sólo hay que leer 2 informaciones; por tanto, la variable B deberá leer, en DATA, de 1 a 2 (FOR B = 1 TO 2).

Líneas 100 a 1200 y 240: los números correspondientes a los datos numéricos se colocan en una tabla alfanumérica. *La variable C* contiene los nombres de los obreros (visualización en 5 líneas).

Líneas 170 a 220: la impresión del contenido de todas las variables puede programarse en el mismo bucle (I), ya que es suficiente imprimir variables que tengan ya un contenido en memoria.

Problema IV — Hoja de salario

Escriba el programa que permita confeccionar la hoja de salarios de 3 empleados.

Los importes se introducen en DATA.

- **Programa** (solución)

```
10 REM : PROGRAMA "SALARIOS"
20 READ N
25 IF N= -2 THEN GOTO 230
30 READ SB,S,R
40 SS=SB*(S/100)
50 SS=INT(SS+.5)
60 IR=SB*R/100
70 IR=INT(IR+.5)
80 LP=SB-SS-IR
90 PRINT TAB(20)"HOJA DE SALARIOS : ";N
100 PRINT TAB(20)"==================="
110 PRINT
120 PRINT "SALARIO BRUTO"; TAB(49)SB
130 PRINT
140 PRINT "COTIZACION S.S."; TAB(50)SS
150 PRINT "RETENCION IMP.RENTA";TAB(50)IR
160 PRINT
170 PRINT TAB(49)"---------"
180 PRINT
190 PRINT "LIQUIDO A PERCIBIR";TAB(49)LP
200 PRINT TAB(49)"============="
210 PRINT:PRINT
220 GOTO 10
230 END
500 DATA 18,211370,6.28,18
510 DATA 19,220260,6.28,22
520 DATA 20,195430,6.28,15
530 DATA -2
```

• **Ejecución**

```
                    HOJA DE SALARIOS :  18
                    ==========================

SALARIO BRUTO                                          211370

COTIZACION S.S.                                         13274
RETENCION IMP.RENTA                                     38047

                                                     --------

LIQUIDO A PERCIBIR                                     160049
                                                     ========

                    HOJA DE SALARIOS :  19
                    ==========================

SALARIO BRUTO                                          220260

COTIZACION S.S.                                         13832
RETENCION IMP.RENTA                                     48457

                                                     --------

LIQUIDO A PERCIBIR                                     157971
                                                     ========

                    HOJA DE SALARIOS :  20
                    ==========================

SALARIO BRUTO                                          195430

COTIZACION S.S.                                         12273
RETENCION IMP.RENTA                                     29315

                                                     --------

LIQUIDO A PERCIBIR                                     153842
                                                     ========
```

• **Observaciones, trucos y consejos**

Leyendo este programa, sin duda el lector se sorprenderá de *hallar en* DATA *un valor raro:* −2. No se trata, como podría creerse, de un error. La línea 2Ø ordena la lectura de las variables contenidas en DATA; estas variables se leerán en el orden de los DATA; es decir, primero los DATA de la línea 5ØØ, luego los de la línea 51Ø y, finalmente, los DATA contenidos en la línea 52Ø.

Ahora bien, al final de la tercera lectura, la línea 2ØØ —que remite a la línea 1Ø— ordena una nueva lectura de variables cuando no hay ninguna línea de DATA correspondiente. Esto provoca la aparición de un mensaje de error, del tipo OUT OF DATA, a pesar de que la ejecución se haga correctamente. Para evitar este mensaje de error, basta añadir al programa una instrucción de salto hacia la instrucción END (línea 23Ø), que se ejecuta cuando una variable toma un valor particular, valor que no podría hallarse en los datos (por ejemplo, un número negativo).

Ejemplo:

— línea 25: IF NI = −2 THEN 23Ø

— línea 23Ø: END

Este valor particular figurará entonces en una línea suplementaria de DATA.

Ejemplo:

— línea 53Ø: DATA −2

La línea 25 implica, pues, un salto hacia la línea 23Ø cuando NI tiene el valor −2 en el momento de la ejecución del programa (línea 2Ø).

Observación: La lectura de la variable NI está aislada en la línea 2Ø. Por esto, basta introducir una sola señal indicadora. Si la lectura de NI se hubiera pedido en la línea 3Ø al mismo tiempo que la lectura de otras variables, *hubiera sido necesario introducir tantas señales indicadoras como variables a leer en DATA*. Por tanto, la línea 53Ø hubiera debido escribirse así:

53Ø DATA −2, −2, −2, −2.

Este valor particular se denomina señal indicadora (fin de fichero).

Problema V — Albarán

Escribir el programa que permita imprimir el albarán de la página siguiente.

• **Programa** (solución)

```
10 A1=142
20 A2=139
30 A3=139
40 A4=138.5
50 A5=141.5
60 A6=A1+A2+A3+A4+A5
70 PRINT":::::::::::::::::::::::::::::::::::::::::"
80 PRINT
90 PRINT"       EMPRESAS ALVARO"
100 PRINT
110 PRINT"  GRAN VIA 47,50006 ZARAGOZA"
120 PRINT
```


10.

```
130 PRINT"::::::::::::::::::::::::::::::::::::::::"
140 PRINT
150 PRINT"                NOTA DE PESOS"
160 PRINT"                -------------"
170 PRINT"ALBARAN 1450"
180 PRINT"FECHA 15 MARZO"
190 PRINT
200 PRINT"NUM.ARTICULO",": NUM. BULTO",": CANTIDAD"
210 PRINT"----------------------------------------"
220 PRINT"7852",":";1,":";A1
230 PRINT,":";2,":";A2
240 PRINT,":";3,":";A3
250 PRINT,":";4,":";A4
260 PRINT,":";5,":";A5
270 PRINT,,": -----"
280 PRINT," PESO TOTAL",":";A6
290 PRINT,,": ======"
```

• **Ejecución**

```
::::::::::::::::::::::::::::::::

        EMPRESAS ALVARO"

     GRAN VIA 47, 50006 ZARAGOZA

::::::::::::::::::::::::::::::::

             NOTA DE PESOS
             -------------

ALBARAN 1450
FECHA 15 MARZO

NUM. ARTICULO  : NUM. BULTO  : CANTIDAD
----------------------------------------

7852           : 1           : 142
               : 2           : 139
               : 3           : 139
               : 4           : 138.5
               : 5           : 141.5
                             :-----
                 PESO TOTAL  : 700
                             :======
```


Problema VI — Factura

Haga la factura correspondiente al albarán anterior. Esta factura ha de dirigirse a D. Juan Casas, Industria, 283, 08026 Barcelona.

El precio unitario (por kilo) es de 2.200 pesetas y el IVA del 12 %.

En la factura figurará el mismo membrete que en el albarán.

• **Programa** (solución)

```
10 A6=700
20 P1=2200
30 T1=A6*P1
40 IV=.12
50 TA=T1*IV
60 T2=T1+TA
70 PRINT"::::::::::::::::::::::::::::::::::"
80 PRINT
85 PRINT"         EMPRESAS ALVARO"
90 PRINT"    GRAN VIA 47, 50006 ZARAGOZA"
100 PRINT
110 PRINT"::::::::::::::::::::::::::::::::::"
120 PRINT
130 PRINT"FACTURA 1450",,"D.JUAN CASAS"
140 PRINT"FECHA 15 MARZO",,"INDUSTRIA 283"
150 PRINT,,"08026 BARCELONA"
160 PRINT
170 PRINT"NUM. ARTICULO",": PESO",": CANTIDAD",":
    PRECIO"
180 PRINT"----------------------------------------"
190 PRINT"7852",":";A6;" K.",":";P1,":";T1
200 PRINT,,"IVA";IV;" %",":";TA
210 PRINT,,,": -----"
220 PRINT,," PRECIO TOTAL",":";T2
230 PRINT,,,": ======"
```

• **Ejecución**

```
::::::::::::::::::::::::::::::::::::

          EMPRESAS ALVARO
     GRAN VIA 47, 50006 ZARAGOZA

::::::::::::::::::::::::::::::::::::

FACTURA 1450            D. JUAN CASAS
FECHA 15 DE MARZO       INDUSTRIA, 283
                        08026 BARCELONA
```

```
-------------------------------------------------------
7852           : 700 K. : 2200        : 1540000
                    IVA .12 %         :  184800
                                      : -------
                  PRECIO TOTAL        : 1724800
                                      : =========
```


ALGUNOS ESTÁNDARES

El Basic y las normas

Ya lo hemos dicho repetidas veces: no hay un lenguaje Basic estándar, sino únicamente numerosos dialectos que son más o menos parecidos al Basic original. Esquematizando esta afirmación podría decirse que cada dialecto Basic está formado por un Basic de base, muy parecido al Basic original —para las funciones esenciales—, y palabras de un dialecto para las funciones sofisticadas. Estas funciones serían la utilización de diskettes, color, impresoras, joysticks, etc.

Como cada fabricante ha dotado sus máquinas con unos perfeccionamientos personales, utilizará un dialecto también personal: este Basic dependerá, pues, esencialmente del hardware. Esto es válido, sobre todo, para los pequeños ordenadores de bolsillo, cuyo dialecto suele ser bastante distinto del de sus hermanos mayores, los ordenadores de oficina.

Si bien es cierto que no existe una verdadera norma universalmente reconocida, conviene citar, sin embargo, una norma de hecho (el Basic Microsoft o MBasic), otra, que parece que va a ser la norma del futuro en el campo de los ordenadores familiares (el MSX Basic), y un ensayo de norma (la norma ANSI).

El MBasic

Este Basic es el más difundido de los dialectos, ya que no depende de una máquina determinada. En efecto, basta disponer de un lector de diskettes para poder cargar inmediatamente este Basic, que funciona en muchísimos sistemas de explotación de

discos (CP/M, MS-DOS, etc.). Creado por la sociedad norteamericana Microsoft, está actualmente en su quinta o sexta versión. Se trata de un Basic extendido con numerosas posibilidades. Son muchas las obras (y muchos los programas) que utilizan este Basic como estándar.

El MSX Basic

Creado por Microsoft en 1983 para un grupo de dieciséis constructores japoneses, constituye la primera tentativa de estandarización aplicada a la informática familiar. El MSX es un derivado del Basic gráfico del IBM PC; se trata, pues, de un lenguaje de nivel profesional adaptado a una utilización doméstica. Una de sus ventajas es que tanto el software como el hardware que cumple la norma MSX es totalmente compatible.

La norma ANSI

Esta sigla es la del importante American National Standards Institute, es decir, el Instituto Nacional Americano para el estudio de los Estándares. Asustado por la multiplicación de los dialectos, este instituto se ha preocupado por la cuestión y ha propuesto cierto número de palabras. Son muchos los fabricantes que se han inspirado en la norma propuesta por el ANSI sin que, sin embargo, la hayan adoptado por completo.



BASIC estándar y dialectos

Basic no dependiente de máquinas

Los Basic enumerados a continuación pueden ser directamente utilizados en muchísimas máquinas; por tanto, pueden definirse en cierto modo como eventuales estándares:

- **CBasic:** Basic semicompilado distribuido por Digital Research y utilizable con el sistema de explotación de discos CP/M. Actualmente hay dos nuevas versiones de este Basic, versiones que se denominan CBasic-80 y CBasic-86 (ver más adelante).

- **OBasic:** Antiguo nombre del MBasic. La compatibilidad entre estos dos Basic es, sin embargo, incompleta.

- **SBasic:** Basic compilado.

- **XBasic:** un Basic redactado especialmente para aplicaciones matemáticas y que dispone de muchas funciones matriciales que habitualmente no están presentes en los pequeños sistemas. XBasic es una creación británica. Este Basic funciona bajo CP/M.

- **XTalBasic:** este Basic permite al usuario competente extender su lenguaje, añadiéndole hasta 64 palabras. También es de creación británica.

- **Comal 80:** se trata de un lenguaje intermedio entre el Basic y el Pascal, desarrollado por una sociedad sueca. Como la mayoría de los Basic, es interpretado. Conviene advertir que este lenguaje

comprende casi todas las instrucciones y comandos del Basic de Microsoft.

CBasic

En su versión estándar, el Basic es un lenguaje interpretado, es decir, cada instrucción es inmediatamente operativa. Esto hace que su enseñanza sea cómoda (los errores se señalan en el momento en que se cometen), pero su utilización está sometida a una cierta lentitud. Por tanto, para conservar al Basic su facilidad de uso dándole, a la vez, la rapidez de sus hermanos mayores (Fortran, Cobol...) se ha pensado en someterlo, a él también, al proceso de **compilación.**

Recordemos que esta expresión significa, simplemente, que el programa, una vez escrito, es traducido «en bloque» a lenguaje máquina. El primer programa —escrito en Basic— se denomina **lenguaje fuente,** y el programa definitivo (= compilado) **lenguaje objeto.**

Existen numerosas versiones compiladas del lenguaje Basic. Algunas compilan un programa previamente interpretado; otras utilizan directamente el compilador.

El CBasic es un lenguaje Basic muy corriente, que emplea directamente el proceso de la compilación. Este Basic, que funciona únicamente con diskettes y bajo CP/M, se emplea habitualmente para aplicaciones comerciales y, en realidad, interesa muy poco al usuario de un ordenador «personal».

Sin embargo, como suele citarse a menudo en revistas y presenta algunas características especiales, nos parece útil, al menos a título informativo, decir algunas palabras sobre él.

En su primera versión (2.08), el CBasic consiste en tres programas:

1. El compilador Basic (denominado CBAS2)
2. El programa de ejecución (CRUN2)
3. Un programa de ayuda (XREF.COM)

El compilador Basic traduce el lenguaje fuente a lenguaje objeto intermedio. Éste es ejecutado por el CRUN2. El programa de ayuda produce un listado de todas las variables empleadas en el lenguaje fuente.

- **Procedimiento a seguir para crear un programa**

El procedimiento a seguir para crear un programa Basic es algo más complicado que en Basic interpretado.

1 — Es necesario crear el programa por medio de un editor de textos cualquiera y darle la extensión .BAS.

2 — A continuación, y por medio del compilador (CBAS2), se compila el programa fuente (que recibe automáticamente la extensión .INT).

3 — El programa podrá ejecutarse por medio del programa de ejecución CRUN2.

- **Características de este Basic**

El CBasic posee la mayoría de las características de los Basic tradicionales, pero hay que destacar las particularidades siguientes:

- no es necesario numerar las líneas, excepto si contienen instrucciones de salto (GOTO, etc.);
- posibilidad de utilizar largas cadenas de caracteres (hasta 31).
- es obligatorio utilizar algunos «blancos» entre las palabras. Así, la expresión FORT=ATOB tiene un significado distinto de la expresión FOR T= A TO B.

GUÍA DE COMPRA DE UN MICROORDENADOR

¿Cómo elegir su microordenador?

Para crear programas en lenguaje Basic o para ejercitarse en el arte de la programación, no es necesario adquirir un ordenador capaz de rivalizar con los de la NASA. Por algunas decenas de miles de pesetas podrá procurarse un pequeño ordenador capaz de satisfacer su pasión y llenar de manera agradable infinidad de veladas.

Los microordenadores que le presentamos en estas páginas se han seleccionado en función de los siguientes criterios:

- precio asequible;
- todos poseen un Basic residente (es decir, implantado en la misma máquina, en ROM);
- están difundidos por todo el mundo y varias decenas (y a veces centenares) de millares de unidades están funcionando ya sin problemas.

Todos estos microordenadores se venden en forma de teclado que contiene el microprocesador y las memorias. Usted no tendrá más que conectarlo a su televisor o, eventualmente, a un monitor externo.

Cada máquina se describirá brevemente, insistiendo en algunos puntos especiales que la diferencian de sus competidoras.

Damos una primordial importancia al teclado, que es uno de los únicos medios de comunicación entre el hombre y el ordenador. Un teclado digital (como el del SINCLAIR ZX 81) es, a la larga, desagradable de utilizar, pero el precio de venta del ordenador explica perfectamente esta desventaja. En cambio, los ordenadores familiares «de la parte alta de la gama» deberían dis-

poner de un teclado normal (esto no es así en el THOMSON o el ATARI 400). Digamos, finalmente, que todos los teclados de los ordenadores familiares son de tipo QWERTY.

Sinclair ZX 81

Esta pequeña máquina, de origen inglés, ha tenido una venta de varios centenares de miles de unidades en todo el mundo. Es el microordenador menos caro que puede encontrarse. Se vende montado (y, por tanto, listo para funcionar) o bien en kit (que no es aconsejable, a menos que usted posea ya algunos conocimientos de electrónica). Dispone de una amplia gama de software y de periféricos.

Sus principales características son:
— ROM: 8k
— RAM: 1k (extensible a 16 K)
— *Teclado:* digital
— *Presentación visual:* 24 líneas de 32 columnas
— *Basic* (residente): bastante potente; no se aparta mucho de las «normas»

La principal ventaja del SINCLAIR ZX 81 es su módico precio y el extenso software de que dispone, también a un precio muy bajo. Para aprender el lenguaje Basic es muy recomendable, pero desde el momento en que desee programar aplicaciones complejas, se verá limitado por sus posibilidades. Su principal inconveniente es un teclado digital muy fatigoso, si bien es posible añadirle un teclado normal pagando un suplemento.

Si piensa ampliar el uso de su microordenador a algo más que no sea el aprendizaje del Basic, no es éste el sistema que le resultará menos caro.

Sinclair ZX Spectrum y Spectrum PLUS

Herederos del ZX 81, los Spectrum poseen, además, un verdadero teclado, color y sonido. Sus precios son sensiblemente más altos, pero tienen la posibilidad de poder utilizar diskettes y de transformarlos en un sistema muy potente.

Sus principales características son:
— ROM: 16 K
— RAM: 16 K (extensible a 48 K). 48 K el Spectrum Plus
— *Teclado:* normal (pero pequeño) el Spectrum y teclado profesional el Spectrum Plus
— *Presentación visual:* 24 líneas de 80 columnas

— *Basic* (residente): potente; no se aparta mucho de las «normas»
— *Posibilidades suplementarias:* sonido y color.

Es un excelente sistema para aprender el lenguaje Basic y las posibilidades sonoras y gráficas (color) de un ordenador. Sin embargo, conviene comparar su precio con el de otros competidores que ofrecen las mismas posibilidades. Finalmente, es posible completar el sistema añadiendo sus muchos periféricos. Digamos, además, que hay disponible un extenso software de alto nivel.

Vic 20 (Commodore)

Este pequeño ordenador ha alcanzado ya una cifra de ventas de más de un millón de unidades y ha estado en cabeza de lista en Estados Unidos. Se trata de un ordenador muy completo que permite aprender el Basic y crear música, sonidos y colores.

Sus principales características son:
— ROM: 20 K (extensible a 28 K)
— RAM: 5 K (extensible a 32 K)
— *Teclado:* normal
— *Presentación visual:* 23 líneas de 22 columnas
— *Basic* (residente): potente; no se aparta mucho de las «normas»

Su principal ventaja está en un teclado muy bien concebido y extenso, en la posibilidad de completar su sistema añadiendo diskettes, así como en sus posibilidades sonoras y gráficas (color).

Finalmente, este sistema permite la adaptación de cartuchos. Los cartuchos son memorias muertas que se conectan directamente al teclado y permiten la inmediata utilización de un programa (es decir, sin lectura previa sobre un soporte cualquiera).

Su precio lo coloca en una ventajosa posición. Digamos, no obstante, que los cartuchos de juego son relativamente caros.

Es muy recomendable para el aprendizaje del lenguaje Basic.

Commodore 64

Hermano mayor del Vic 20, el Commodore 64 ha logrado una gran difusión. Sus posibilidades de ampliación son considerables y existe un software muy completo para él.

Sus principales características son:
— ROM: 20 K
— RAM: 64 K
— *Teclado:* normal
— *Presentación visual:* 25 líneas de 40 columnas

Tandy Color

Uno de los primeros ordenadores de la parte baja de la gama, capaz de generar sonidos y color. Puede admitir numerosos periféricos y posee una extensa biblioteca de software.

Sus principales características son:

— ROM: 8 K
— RAM: 16 a 32 K
— *Teclado:* normal
— *Presentación visual:* 16 líneas de 32 caracteres
— *Basic* (residente): potente; corresponde a las «normas» Microsoft

La principal ventaja de este sistema es la de disponer de un excelente teclado y de un Basic Microsoft. Genera también color y sonidos.

Su principal inconveniente es su mala presentación visual: 16 líneas es poco, sobre todo si el programa es relativamente largo.

Como para el VIC 20, es posible conectar cartuchos ROM y, de este modo, disponer de un software «listo para su uso». Su precio es relativamente alto.

Sony Hit Bit 55

Su principal atractivo es que cumple el estándar MSX, lo que permite intercambiar y acoplar el distinto software y hardware que ofrecen las demás marcas MSX.

Sus principales características son:

— ROM: 32 K
— RAM: 16 K (extensible a 32 K)
— *Teclado:* profesional

Atari 400

Se trata más bien de una consola de juegos que de un verdadero ordenador, a pesar de que dispone de un lenguaje Basic.

Sus principales características son:

— ROM: 16 K
— RAM: 16 K
— *Teclado:* digital
— *Presentación visual:* 24 líneas de 40 caracteres
— *Basic* (residente): bastante apartado de las «normas»

Su ventaja principal está en una extensa biblioteca de software de juegos y en una presentación visual de calidad (sobre todo por la alta resolución gráfica). Ordenador de juegos, posee, naturalmente, las conexiones necesarias para los cartuchos y las palancas de juego.

No es recomendable para el aprendizaje del lenguaje Basic, especialmente teniendo en cuenta su precio, bastante elevado.

Como desventajas, digamos que la principal es su teclado numérico, cosa realmente sorprendente en un ordenador de su precio.

Dragon 32

Es un buen representante del acertado criterio británico. La idea de principio de este microordenador es la construcción de un sistema fácil de utilizar y, a la vez, robusto (es el «Premio Fisher» de los ordenadores).

Sus principales características son:

— ROM: 16 K
— RAM: 32 K (extensible a 64 K)
— *Teclado:* normal y robusto
— *Presentación visual:* 16 líneas de 32 caracteres
— *Basic* (residente): potente y de «normas» Microsoft

Este microordenador es muy parecido al Tandy Color, con el cual es compatible a nivel de cartuchos de juegos. Como el Tandy, genera también color y sonidos.

Su principal desventaja es una presentación visual muy pobre (16 líneas de 32 caracteres). Su precio lo sitúa entre los de la parte alta de la gama.

Es muy recomendable para el aprendizaje del lenguaje Basic.

Oric 1

Forma parte de la última invasión británica. Muy atractivo en numerosos aspectos, ofrece varias características nuevas como, por ejemplo, posibilidades telemáticas.

Sus principales características son:

— ROM: 16 K
— RAM: 16 K (extensible a 64 K)
— *Teclado:* normal
— *Presentación visual:* 28 líneas de 40 caracteres
— *Basic* (residente): potente y muy cercano a las «normas»

Este micro de la nueva generación ofrece, bajo un volumen muy compacto, numerosas posibilidades. Concebido más bien como un sistema «multiusuarios» (familiar, pero también para ne-

gocios o educación), no posee conexión para cartuchos, pero en cambio permite conectar todos los periféricos (incluido el modem, para la comunicación a distancia). Es un micro que tiene también posibilidades sonoras y gráficas (color) muy interesantes. Su precio es muy atractivo.

Recomendable para el aprendizaje del lenguaje Basic.

Thomson T0 7

Presenta una particularidad destacable (**un lápiz óptico,** que permite utilizar la pantalla como periférico de entrada) y otra que lo es menos (**un teclado de membranas).**

Sus principales características son:

— ROM: 6 K
— RAM: 22 K
— *Teclado:* de membranas
— *Basic:* (residente): potente; corresponde a las «normas» Microsoft

Este micro es muy recomendable para el aprendizaje del Basic, si no fuera por su teclado, de uso desagradable (su principal defecto).

En el palmarés de las ventajas, destaquemos las posibilidades sonoras y gráficas, así como el lápiz óptico. El precio es relativamente elevado, teniendo en cuenta la desventaja del teclado. Mencionemos también la posibilidad de utilización de cartuchos de memoria muerta.

DICCIONARIOS

Los dos diccionarios que siguen, orientados especialmente hacia los lenguajes informáticos y la programación en Basic, le ayudarán a comprender el argot de los informáticos y, quizás, a comprar con más acierto el ordenador que le interese, gracias a un diálogo más fácil con el vendedor.

El primer diccionaraio reproduce los principales términos informáticos utilizados habitualmente en informática familiar. El segundo está dedicado a los principales vocablos del lenguaje Basic. Le ayudará a encontrar con rapidez la función que la palabra representa.

Recordemos que las palabras del lenguaje Basic pueden dividirse en cuatro categorías:

1. Las palabras de **comando:** ordenan al sistema que efectúe determinadas operaciones en el programa (puesta en marcha: RUN, listo: LIST, etc.).

2. Las palabras de **declaración:** son palabras que constituyen el programa propiamente dicho; se trata, en realidad, de las instrucciones del programa: LET, GOTO, etc.

3. Las palabras de **funciones:** son palabras para recurrir a pequeños programas internos como, por ejemplo, el cálculo del seno, coseno, etc.

4. Los vocablos **operadores:** se trata, generalmente, de símbolos que controlan la naturaleza de la operación a efectuar; se distinguen los operadores matemáticos (+, —, ...), los operadores lógicos (NOT, AND,...) y los operadores de comparación (mayor que, etc.).

Diccionario de vocablos informáticos

Ada

Lenguaje informático creado por el departamento de defensa de Estados Unidos. Lo ideó un informático francés, J. Ichbiach, que le dio el nombre de una colaboradora de Babbage, matemático inglés, considerado el primer programador del mundo.

Algol

Acrónimo de *Algorithmic Language*. Es un lenguaje de programación científica concebido para satisfacer tanto a matemáticos como a informáticos. De ALGOL se han hecho varias versiones, la última de las cuales se denomina ALGOL 68.

Algoritmo

Descripción de la realización de un suceso cualquiera por medio de fórmulas que describen cada acción elemental. Es la «receta» para llegar fácilmente al resultado que se pretende.

APL

Sigla de *A Programming Language*. Lenguaje de programación conciso y denso. Se aparta mucho de los demás lenguajes de

programación por su estructura (sintaxis y semántica) y por la utilización de una notación original.

ASCII

Sigla de American Standard Code for Information Interchange. Es un código que se emplea para transferir las informaciones.

Asignación

Atribución de un valor a una variable por medio de un signo especial, o, en Basic, de la palabra clave LET.

Basic

Acrónimo de Beginner's All Purpose Symbolic Instruction Code. Este lenguaje de programación permite utilizar muy rápidamente las posibilidades de un ordenador. Desgraciadamente, y a diferencia de lo que suele ser costumbre en otros lenguajes informáticos, no existe un Basic estándar, sino solamente numerosos «dialectos» más o menos parecidos al Basic original.

Binario

Es el único «lenguaje» que comprende el ordenador: es el *lenguaje máquina*, sistema de comunicación compuesto únicamente por las cifras Ø y 1.

Bit

Contracción de Binary Digit. Es la unidad de información más pequeña: Ø ó 1. Si para comunicar con el ordenador se emplean secuencias (o palabras) de 8 bits, se hablará de un ordenador de 8 bits. Una palabra de 8 bits se denomina *byte*.

Bucle

Conjunto de instrucciones repetido varias veces, hasta que se ha realizado una condición de salida de bucle (o de paro de

bucle). En Basic, un bucle se programa mediante la instrucción FOR... NEXT o de la instrucción GOTO.

Buffer

Pequeña memoria intermedia entre un ordenador y sus periféricos.

Byte

Palabra inglesa que designa un conjunto de 8 bits. Con 8 bits, en lenguaje binario, se pueden hacer 256 combinaciones distintas (2 exponente 8).

Cabecera

Informaciones colocadas al principio de un fichero en soporte magnético y que hacen posible el reconocimiento. Sinónimos: *etiqueta, nombre del programa.*

Cadena

Conjunto de datos de una misma naturaleza colocados uno a continuación de otro. Con el nombre de cadena de caracteres se designa una serie de caracteres alfanuméricos. Son varias las instrucciones de Basic que sirven para crear y modificar cadenas de caracteres. La mayoría de los dialectos emplean el símbolo «dólar» ($) para designar una variable compuesta por una cadena de caracteres. Por ejemplo A$ = «Buenos días».

Campo

Traducción de la palabra inglesa *field*, que designa la parte de un registro. Según el contexto, se hablará también de *zona, entrada* o *rúbrica.*

Carga

Puesta en memoria central de datos o de un programa contenidos en memoria auxiliar (casete, diskette...). En Basic se emplea habitualmente el comando LOAD (de *loading*, palabra inglesa que significa carga). Algunas veces, para designar la carga a partir de una casete, el comando se convierte en CLOAD.

Catálogo

Lista de los programas contenidos en un diskette. Se utiliza también la palabra inglesa *directory.*

Cero

No hay que confundirlo con la letra O. Para evitar confusiones (que debería corregir el usuario, no el ordenador), *la cifra cero va siempre barrada: Ø.*

Cinta magnética

Es la memoria externa (o memoria de masa) más elemental. En microinformática familiar, se utiliza corrientemente la casete magnética, que no es más que una casete audio de buena calidad.
Soporte de información barato, la cinta magnética solamente permite un acceso secuencial, es decir, secuencia tras secuencia.

Cobol

Acrónimo de Common Business Oriented Language. Como su nombre indica, es un lenguaje especialmente orientado a la gestión. Existen varias versiones. A pesar de la aparición de otros lenguajes, el COBOL sigue siendo el lenguaje mejor adaptado a los problemas de gestión.

Coma

En lenguaje Basic tiene varios usos. El más frecuente lo encuentra en la instrucción PRINT, en la que provoca tabulaciones automáticas.

Comando

Palabra de lenguaje que ordena al ordenador que efectúe una acción en un programa. Ejemplos: LIST, RUN, NEW... En algu-

nos Basic, el comando puede formar parte del programa, convirtiéndose en una declaración.

Comillas

En lenguaje Basic, delimitan una cadena de caracteres.

Compilador

Programa que permite traducir otro programa escrito en lenguaje evolucionado, como por ejemplo el Basic, en lenguaje máquina. Un compilador traduce el programa en una sola vez, en tanto que un intérprete lo traduce instrucción por instrucción. La versión del programa no compilada se denomina *programa fuente*, y la versión compilada *programa objeto*.

Consola

Con este término se designa el conjunto formado por el teclado y la consola.

Copia de seguridad

Es la copia que se hace de un programa para prevenir la posible pérdida o destrucción de éste.

Cursor

Señal en la pantalla que indica el lugar del próximo carácter. Se trata de una señal luminosa que, a menudo, es parpadeante. Algunas veces es posible desplazar el cursor por medio de teclas de función marcadas con una flecha.

Chip

Palabra inglesa con la que se designa la pastilla que contiene un circuito integrado; es un término muy utilizado en microinformática.

Datos

Valores necesarios para que el ordenador pueda efectuar el programa. A veces, los datos están ya contenidos en el programa Basic en forma de una declaración DATA. Estos datos son leídos por una instrucción READ.

Decrementar

Disminuir el valor de un contador en una cantidad fijada, por ejemplo, en un bucle de programa.

Depuración

Traducción (aproximada) del inglés *debugging*. Búsqueda de los errores (o *«bugs»)* de un programa.

Diagrama de flujo

Representación, con ayuda de símbolos convencionales, de las distintas etapas necesarias para la resolución de un problema.

Dimensión

Declaración Basic que fija el número de elementos que comprende una tabla, así como su organización (una o más dimensiones). En la mayoría de los Basic, esta declaración se denomina, simplemente, DIM.

DOS (véase SED).

EAO (Enseñanza Asistida por Ordenador)

En inglés, CAI (Computer Assisted Instruction). Nuevo método pedagógico que utiliza las posibilidades gráficas, sonoras y repetitivas del ordenador para aprender una lengua, un oficio o... el lenguaje Basic.

Editor

Programa de servicio concebido para facilitar la entrada o la modificación de textos. El editor permite posicionar el cursor en cualquier lugar del texto.

En línea (en inglés, *On-line*)

Dícese de un periférico directamente conectado al ordenador. Este término es el opuesto de *Off-line*.

Ensamblador

Lenguaje informático muy próximo al lenguaje máquina. Cada microprocesador posee su propio lenguaje de ensamblamiento, lo que hace que el aprendizaje de este último sea útil únicamente para programar en un solo ordenador.

Entero

Número sin parte fraccionaria.

Entradas-Salidas (E/S) (en inglés, *Input/Output*)

Conjunto de operaciones que permiten la unión de un ordenador a sus periféricos.

Espacio

Intervalo entre caracteres. Algunos lenguajes Basic aceptan intervalos; otros, los rechazan (el programa no funciona y el error no se pone de manifiesto...). Ejemplo: GOTO o GO TO. El espacio se obtiene pulsando la barra de espacios.

Estándar

Se dice de lo que está de acuerdo con la norma. Algunos lenguajes informáticos tienen estándares, en el sentido de que todas las versiones disponibles para los ordenadores son idénticas.

Desgraciadamente, no es éste el caso del lenguaje Basic, del que sólo existen «dialectos» derivados, por añadidura, del Basic primitivo. Sin embargo, el organismo norteamericano de normalización (ANSI) ha propuesto cierta normalización mínima, pero sin demasiado éxito. También Microsoft ha creado, para un grupo de dieciséis constructores japoneses, un estándar muy atractivo, el MSX.

Fichero (en inglés, *file*)

Conjunto estructurado de instrucciones almacenado en un soporte magnético. Al principio, el fichero designaba un conjunto de fichas, pero ahora esta palabra se utiliza para todo un conjunto de instrucciones que constituyen un bloque lógico. Así pues, todo programa es un fichero.

Floppy Disk

Expresión inglesa para designar el diskette.

Formateado (en inglés, *formatting*)

Operación que consiste en preparar un diskette virgen para darle un formato utilizable por el ordenador.

Forth

Abreviatura de *fourth* (cuarta generación). Es un lenguaje informático muy potente que entra en directa competencia con el Basic. Requiere poca memoria, permite la programación estructurada y el añadido de instrucciones personales, y es diez veces más rápido que el Basic.

Fortran (FORmula TRANslation system)

Es el «padre» del lenguaje Basic. Lenguaje informático compilado y ya antiguo, ha visto anunciada su desaparición muchas veces. Sin embargo, todos los ordenadores disponen de una versión de este lenguaje, todavía muy empleado. La última normalización data de 1977 (FORTRAN V o FORTRAN 77).

Fuente

Dícese de un programa que ha de compilarse antes de su utilización (pasa a ser, entonces, programa-objeto).

Fusión *(merging)*

Reunión de varios ficheros en uno solo.

Generador de caracteres

Es el chip responsable de la formación de los caracteres en la pantalla o en la impresora matricial.

Hard copy

Copia del texto que aparece en pantalla efectuada con impresora.

Hardware

Término opuesto a software. Designa todos los elementos físicos empleados en un sistema informático. Es, por consiguiente, la parte tangible, material, es decir, todo lo que no es el software: mueble, elementos mecánicos, magnetófono, etc.

Impresora (en inglés, *printer*)

Periférico de salida que permite obtener sobre papel el contenido de la pantalla, de la memoria o los resultados de los programas. En informática individual, se utilizan, principalmente, impresoras matriciales, cuya velocidad de escritura varía entre 30 y 100 caracteres por segundo. Para conectar la impresora al ordenador es indispensable disponer de un interfaz.

Incremento

Valor constante añadido sistemáticamente a un contador.

Instrucción (en inglés, *statement*)

Operación que consiste en dar una orden a un ordenador. Un programa está formado por una serie de operaciones que hay que efectuar dentro de un orden lógico. Se distinguen cinco grandes clases de instrucciones:

1. Instrucciones aritméticas.
2. Instrucciones lógicas.
3. Instrucciones de movimiento.
4. Instrucciones de salto.
5. Instrucciones de entrada/salida.

Interfaz

Elemento (de hardware o de software) intermediario entre el ordenador y sus periféricos. El interfaz permite la comunicación en modo paralelo (los 8 bits se envían al mismo tiempo) o en modo serie (los bits se envían uno después de otro).

Intérprete

Programa que efectúa la traducción y la ejecución de las instrucciones línea a línea. El lenguaje Basic es un lenguaje interpretado (lo que permite probar cada línea), contrariamente al lenguaje Fortran, que es compilado (el programa se traduce de una sola vez).

ISO (International Standards Organization)

Organismo internacional cuya misión consiste en crear normas en los países industriales. A su vez, cada país posee un organismo nacional de estandarización: en Francia, AFNOR; en Estados Unidos, ANSI, etc. Para el lenguaje Basic existe una norma ANSI (que corresponde al Basic mínimo), que, sin embargo, es poco respetada.

Joystick

Dispositivo que permite controlar los juegos en la pantalla sin utilizar el teclado.

Juego de instrucción

Conjunto de las instrucciones que un ordenador es capaz de ejecutar inmediatamente.

Justificación

Alineamiento de los caracteres de una línea en el margen de la izquierda (tal como se hace en los textos impresos) o en el de la derecha.

Kcs (Standard Kansas City)

Estándar de velocidad para el registro de datos en una casete. Es, pues, la norma de registro y de lectura de las informaciones que circulan entre el ordenador y el lector de casetes.

Kilobyte

Abreviado: K. Representa el tamaño de la memoria disponible. 1 K = 1.024 bytes (2 a la potencia 10).

Lanzamiento

Operación que provoca el principio de una acción. En lenguaje Basic, el lanzamiento de un programa lo efectúa el comando RUN.

Lector

Periférico de entrada que permite la lectura de los datos. Existen diversos lectores en función del soporte de la información: casete, diskette, lápiz óptico, etc.

Lectura

Es la acción de transcribir los datos en forma explotable por el ordenador. La lectura de los datos se efectúa, generalmente, a partir de un soporte magnético (casete o diskete), o directamente, al teclado.

Lenguaje evolucionado (en inglés, *advanced language*)

Lenguaje informático parecido a los lenguajes humanos. Se opone al lenguaje máquina, que corresponde al único lenguaje que el ordenador comprende (la corriente pasa o no pasa). El Basic es el lenguaje evolucionado más utilizado en microinformática. Otros lenguajes evolucionados, como el Cobol, el Fortran, el Algol, etc., se utilizan también en microinformática.

Historia, en fechas, de los principales lenguajes: 1956: Fortran; 1958: Algol; 1960: Cobol, Lisp; 1965: Basic, PL/1; 1969: Pascal; 1971; Logo; 1979: Ada.

LISP (LISt Processing)

Lenguaje de programación de muy alto nivel y de estudio bastante complejo.

Listado

Impresión del programa sobre papel. Por extensión, este término designa también el papel impreso.

LOGO

Lenguaje de programación ideado en los años 1970, pero de implantación muy reciente en microordenadores. Parece estar muy bien adaptado a la enseñanza.

LSE (Lenguaje Simbólico de Enseñanza)

Lenguaje ideado en Francia en 1974 y utilizado, principalmente, en los institutos y universidades de dicho país. No se emplea, sin embargo, en aplicaciones profesionales, por lo que no alcanzará nunca una gran difusión.

Mantenimiento

Conjunto de operaciones que tienen por objeto mantener el hardware o el software en buen estado de funcionamiento. Los microordenadores necesitan muy poco mantenimiento.

Memoria

Parte del sistema informático que permite el registro, la conservación y la restitución de las informaciones. Hay que distinguir entre la memoria central y las memorias auxiliares.

Memoria auxiliar

Memoria utilizada como complemento de la memoria central. Se la conoce también como *memoria de masa*. En microinformática, las memorias auxiliares utilizadas son casetes o diskettes.

Memoria central

Es la memoria interna del ordenador. Cuando sólo puede ser leída se dice que es memoria muerta (o ROM); cuando también puede escribirse en ella se dice que es memoria viva (o RAM).

Memoria muerta (véase ROM).

Memoria viva (véase RAM).

Menú

Lista de los comandos u opciones ofrecidos en un programa. Un programa bien hecho utiliza numerosos menús, lo que permite un trabajo conversacional (el usuario sólo ha de escoger la opción que le interesa).

Microprocesador

Circuito integrado de gran capacidad concebido para el tratamiento y gestión de los datos. Normalmente, los microprocesadores son de 8 o de 16 bits.

Monitor

En informática, esta palabra tiene varios significados, que es preciso no confundir:

1. Pantalla de visualización de gran calidad (= hardware)
2. Programa de explotación (= software)

ON-LINE (véase **En línea**).

Package

Se trata de un programa —acompañado de una completa documentación— que puede interesar a numerosos usuarios. Es, pues, el *prêt-à-porter* de la informática.

Palabra

En general, designa un conjunto de bits que ocupan una sola posición en memoria del ordenador. Los ordenadores de 8 bits tienen palabras de 8 bits. Desde el punto de vista memoria, una palabra equivale a un byte.

En lenguaje Basic, designa cadenas de caracteres que son reconocidas por el lenguaje como medios de comunicación.

Pantalla (en inglés, *screen* o *display*)

El más importante periférico de salida. En microinformática se emplean pantallas de cristales líquidos, pantallas de televisión o monitores.

Pascal

Lenguaje de programación concebido de manera que favorece la programación estructurada. Este lenguaje, heredero del Algol, está obteniendo un éxito cada vez mayor. Ha inspirado otros lenguajes, entre ellos el Ada.

Periférico

Todo lo que no es el microprocesador propiamente dicho y sus elementos-satélite más inmediatos (memorias...) es un periférico.

Los principales periféricos utilizados en microinformática son la pantalla, el teclado, las casetes (o diskettes) y la impresora.

Los periféricos se clasifican en elementos de entrada (para

introducir datos en el ordenador; por ejemplo: el teclado), elementos de salida (para comunicar los resultados; por ejemplo: la pantalla), y elementos de almacenamiento (para registrar los datos; por ejemplo: casetes).

Pilot

Lenguaje informático especialmente destinado a la enseñanza. Permite crear software didáctico sin ningún conocimiento de informática.

Pixel

Es la más pequeña superficie homogénea de una imagen. En cierto sentido, es el equivalente de un punto.

PL/1 (Programming Language 1)

Lenguaje de programación de muy alto nivel, concebido por IBM en 1963 y con unas posibilidades muy parecidas a los principales lenguajes de gestión y científicos existentes (Fortran, Algol y Cobol).

Portabilidad

Este neologismo se aplica a un software que puede transponerse fácilmente de un ordenador a otro. La «oportunidad» de un software es una importante cualidad, ya que le asegura una enorme difusión. Los programas escritos en lenguaje Basic tienen, generalmente, una gran portabilidad, ya que bastan algunas adaptaciones menores para que puedan ser ejecutados por cualquier ordenador que disponga de este lenguaje.

Programa

Conjunto de instrucciones que ordenan al ordenador la ejecución de ciertas tareas en el marco de la realización de un objetivo determinado (cálculo de impuestos, juego, etc.). El lenguaje Basic permite escribir programas.

Programa didáctico

Como su nombre indica, es un programa destinado a la enseñanza.

PSI

Pequeño Sistema Individual. Con esta expresión se designa también a los ordenadores familiares.

Puerta

Punto de entrada y de salida del microordenador programable con software.

Puesta a cero

Puesta a cero de todas las variables y borrado de la pantalla. Cuando la ejecución de un programa queda bloqueada, se efectúa una puesta a cero general (reinicialización) pulsando la tecla «reset».

Qwerty

Las seis primera letras de los teclados americanos. En algunos países de Europa se utiliza un teclado cuyas seis primeras letras son AZERTY y que dispone de caracteres acentuados. No obstante, la disposición de las letras tiene poca importancia para programar en lenguaje Basic, máxime teniendo en cuenta que éste no emplea caracteres acentuados.

RAM (Random Acces Memory)

Es la memoria viva que permite lectura y escritura. Se opone a la memoria muerta (o ROM), que sólo permite la lectura.

Randomizar

Anglicismo. Significa hacer aleatorio. En Basic se dispone de una instrucción (RANDOM) que «mezcla» los números y los da

en «desorden» en la salida. Es una especie de presentación de los números al azar.

Registro

Pequeña memoria destinada a almacenar de manera provisional cierto número de informaciones.

Registro de un fichero (en inglés, *record*)

Bloque de informaciones. En un fichero, cada bloque de información se denomina registro. Es también la escritura de las informaciones sobre un soporte magnético cualquiera.

Reservada (palabra)

Se llama palabra reservada a una palabra clave que el programa reconoce y que no puede ser utilizada como dato.

Residente

Dícese del programa que está permanentemente en memoria. Es el caso del programa monitor y, algunas veces, del programa Basic en los pequeños ordenadores.

ROM (Read Only Memory)

Memoria muerta. En oposición a la memoria viva (o RAM), sólo permite la lectura.

Rutina

Es una serie de instrucciones ejecutables a partir de cualquier punto del programa. En Basic, las rutinas se colocan al final del programa y se accede a ellas mediante la instrucción GOTO y GOSUB.

Scrolling

Movimiento continuo (vertical u horizontal) de los signos presentados en pantalla. Es como si un rodillo, desde detrás de la pantalla, fuera presentándolos: los nuevos datos aparecen mientras los antiguos desaparecen.

SED

Sistema de explotación de diskettes. Si el sistema no posee diskettes, se empleará el término monitor e, incluso, el de supervisor.

Secuencial

Dícese de lo que se realiza secuencia tras secuencia. La búsqueda secuencial de datos se realiza leyendo dichos datos unos después de otros, tal como se van leyendo. Es el único acceso posible con un lector de casetes. El acceso secuencial se opone al acceso directo, que permite obtener inmediatamente la información buscada sin tener que releer las informaciones que las anteceden (este acceso sólo es posible con un lector de diskettes).

Serie

Transmisión de las informaciones entre el ordenador y sus periféricos según el modo serie, es decir, un bit después de otro. El modo serie se opone al modo paralelo, en el que los 8 bits de un byte se transmiten al mismo tiempo.

Sintaxis

Son las reglas que rigen el orden de las palabras y la construcción de las frases. Todos los lenguajes informáticos poseen una sintaxis que es obligado respetar.

Sistema de explotación

Es el software de base de todo ordenador. Se denomina también programa monitor o de DOS

Software

Es la parte no mecánica del ordenador: los programas. Existen programas de explotación y programas de aplicación. Los lenguajes informáticos son también software. Un software estándar se conoce con el nombre de **package.**

Tecla de función

Tecla especial de un teclado que sirve para la ejecución inmediata de una función.

Teclado

En inglés, *keyboard.* Primer periférico de entrada, puede seguir la norma AZERTY (utilizada en algunos países europeos) o la norma QWERTY (americana), según cuáles sean sus seis primeras letras. Además del teclado alfanumérico, como el de las máquinas de escribir, el teclado de un ordenador incluye también las teclas de función y las de comando, y algunas veces un teclado numérico separado. Por tanto, no es raro que algunos teclados de ordenador dispongan de más de 100 teclas distintas.

Teletipo

Marca de teleimpresora. Este nombre de marca se ha convertido casi en un nombre corriente para designar una impresora conectada a un ordenador.

Terminal

Sistema que suele estar formado por un teclado y una pantalla, conectado todo ello a un ordenador central. Cuando un microordenador está conectado a un ordenador central, se convierte en un terminal inteligente.

Tiempo compartido (en inglés, *time sharing*)

El tiempo total de un ordenador central es compartido entre varios usuarios, que disponen de un terminal conectado a este ordenador. La velocidad de ejecución del ordenador es considerable, por lo que el usuario tiene la impresión de ser el único que trabaja con el ordenador central. En principio, el lenguaje Basic se pensó para la utilización de un ordenador a tiempo compartido por no informáticos.

Traducción

Operación que consiste en hacer pasar un programa de un lenguaje a otro.

Si la traducción se hace de una lengua simbólica (Basic, Cobol, etcétera) al lenguaje máquina, entonces el traductor se llama **intérprete** o **compilador** (ver estos dos vocablos).

Si la traducción se hace de un lenguaje de ensamblador al lenguaje máquina, el traductor es entonces un **ensamblador.**

Si la traducción se hace de un lenguaje evolucionado a otro lenguaje evolucionado, el traductor será un **emulador.**

UAL

Unidad Aritmética y Lógica. Es la parte del microprocesador que trata las informaciones matemáticas y lógicas.

Unidad central (en inglés, *CPU* o *Central Processing Unit*)

Es la parte principal del ordenador, es decir, el microprocesador y las memorias directas.

Utilitario

Dícese del programa que facilita la ejecución de los programas y la explotación del sistema. Generalmente, este programa forma parte del programa monitor o DOS.

Variable

Todo dato no fijo y que evoluciona a lo largo del tratamiento de las informaciones y de la ejecución del programa. En lenguaje

Basic se distinguen, principalmente, *variables numéricas* (únicamente cifras) y *variables alfanuméricas* (cifras y letras).

Volcado de la memoria (en inglés, *dump*)

Registro del contenido de la memoria en un soporte papel con el fin de salvaguardarlo o poder buscar los errores.

V24

Denominación empleada a veces para la salida serie.

Zona (en inglés, *field*)

Posición en la que están dispuestas las informaciones. Se emplea también la palabra *campo* (o *rúbrica*). En una ficha descriptiva, por ejemplo, cada característica (nombre, apellidos, dirección, etc.) corresponde a una zona y como tal se lee.

Las principales palabras clave del Basic

A. (véase ABS).

ABS (Función).
Del inglés *ABSolute*.
Esta función determina el valor absoluto de un número o de una variable numérica. Dicho de otro modo: no tiene en cuenta el signo del número.
Ejemplo: PRINT ABS (—1Ø) (en la pantalla aparecerá 1Ø).
Algunos sistemas utilizan la abreviatura A.

AUTO (Comando).
Del inglés *AUTOmatic*.
Esta palabra provoca *una numeración automática de las líneas* del programa. Si no se precisa nada, la numeración se hará de 1Ø en 1Ø.
Ejemplos:
AUTO: (numeración de 1Ø en 1Ø: 1Ø, 2Ø, 3Ø...);
AUTO 7, 2 (numeración a partir de la línea 7 por pasos de 2: 7, 9, 11...).
Para detener este comando se pulsa (según el sistema) la tecla BREAK, RETURN o CTRL C.

ASC (Función)
Del inglés *American Standard Code for Information*.
Esta función *transforma una variable-cadena o un carácter en su código decimal* ASCII.

Ejemplo: PRINT ASC («A») (en la pantalla aparecerá 65, que es el valor ASCII del carácter («A»).

El argumento ha de estar entre paréntesis, y si se trata de una variable-cadena, entre comillas.

Algunos sistemas utilizan la palabra ASCII.

ASCII (véase ASC).

BREAK (Instrucción)

Del inglés *break* = paro.

Provoca *el paro de la ejecución de un programa en el número de línea indicado.*

En algunos ordenadores, la tecla BREAK provoca el paro del programa.

Téngase en cuenta que BREAK *conserva el contenido de las variables.*

Ejemplo: 1∅ BREAK 5∅ (detiene el programa antes de la línea 5∅).

C. (véase CONT).

CLEAR (véase CLS).

CLOAD (véase LOAD).

CLS (Declaración/comando)

Del inglés *Clear Screen.*

Borrado de la pantalla.

Este comando puede también emplearse *en modo directo* por medio de la tecla CLEAR, presente en muchos teclados de microordenadores.

En muchos ordenadores, este comando es ejecutado por la instrucción HOME.

CO (véase CONT).

CON (véase CONT).

CONT (Comando)

Del inglés *continue.*

Este comando vuelve a poner en marcha un programa después de haber sido interrumpido por STOP o BREAK.

La ventaja de este comando es que *conserva las variables*, a diferencia de RUN, que pone las variables a cero.

Algunos aparatos reconocen este comando gracias a una de estas palabras: CON, CO o C.

CHR$ (Función)

Del inglés *CHaRacter)*

Esta función *genera el carácter* (o la función) *correspondiente al código* ASCII colocado entre paréntesis.

Ejemplo: PRINT CHR$ (65) (en la pantalla aparece el carácter A).

El código puede estar representado por un valor numérico, pero también puede estarlo por una variable numérica o una expresión.

CSAVE (véase SAVE).

D (Véase DATA).

DAT (véase DATA).

DATA (Declaración)

Del inglés *Data* = datos.

Utilizado siempre con la instrucción READ. *Lista de datos* que serán leídos por la instrucción READ.

Los datos han de estar *separados por una coma.*

Ejemplo: DATA VIRGATCHIK, JUVENTUD, 1986.

Algunos sistemas utilizan las palabras DAT o D.

DEFDBL (Declaración)

Del inglés *DEFine* = definir, y *DouBLe* = doble.

Esta declaración sirve para *precisar que una o varias variables son del tipo de doble precisión.*

Esta declaración *va siempre al principio del programa.* Hay que tener en cuenta que consume mucha memoria y sólo ha de emplearse en caso de necesidad.

Ejemplo:

DEFDBL A (la variable A será en doble precisión).

DEFDBL A-D (todas las variables identificadas por las letras comprendidas entre A y D estarán en doble precisión).

DEF FN (Declaración)

Del inglés *Define* = definir, y FN = función.

Declaración que *crea una nueva función* utilizable posteriormente.

La estructura de esta declaración *emplea dos variables:* la primera (en nuestro ejemplo: A) caracteriza esta declaración (en rea-

lidad, pueden crearse varias); la segunda (aquí: B) está entre paréntesis y volverá a aparecer a la derecha del signo de asignación (=).

Ejemplo: DEF FNA (B) = B * B. La función FNA tiene la misión de elevar al cuadrado la variable B.

DEFINT (Declaración)

Del inglés *DEFine* = definir, e *INTeger* = número entero.

Esta declaración *precisa que una o varias variables son de tipo entero* (es decir, no tiene en cuenta las cifras que van después de la coma).

Ejemplo: DEFINT A (la variable A quedará siempre redondeada a la cifra entera).

Esta instrucción permite ganar lugar en la memoria y efectuar cálculos más rápidos.

DEFSNG (Declaración)

Del inglés *DEFine* = definir, y *SINGle* = simple.

Esta declaración *precisa que las variables están en «simple precisión».* Anula la función DEFINT.

Ejemplo: DEFSNG A (los resultados de la variable A estarán en simple precisión).

DEFSTR (Declaración)

Del inglés *DEFine* = definir, y *STRing* = cadena.

Esta declaración *precisa que una o más variables son variables-cadenas.*

Ejemplo: DEFSTR A (la variable A será considerada como una variable cadena).

El programa considerará esta variable exactamente igual que si estuviera seguida del signo dólar.

Esta declaración se coloca al principio del programa.

DEL *(véase DELETE).*

DELETE (Comando)

Del inglés *Delete* = suprimir.

Borra una o varias líneas de un programa. La o las líneas a borrar han de precisarse.

Este comando lleva a veces el nombre de DEL.

Ejemplos:

DELETE 30 (borra de la memoria viva la línea 30).

DELETE 30-100 (borra de la memoria todas las líneas de 30 a 100.

Para suprimir todas las líneas de un programa se emplea el comando NEW.

DIM (Declaración)

Del inglés *DIMension* = dimensión.

Esta declaración *fija el número de elementos que comprenderá una tabla* y, por tanto, *la dimensión de la memoria a reservar.*

Algunos Basic admiten también que se definan tablas de más de una dimensión.

Ejemplos: DIM A (10) (la tabla A tendrá 11 elementos). DIM B (10,15) (la tabla B tendrá 11 × 16 elementos).

E. (véase END).

EDIT (Comando)

Del inglés *edit* = modificar.

Este comando ha de ir seguido de un número de línea. La línea indicada es visualizada y *el ordenador se pone en modo editor, permitiendo la corrección de errores.* Por consiguiente, este comando evita volver a escribir una línea que contenga un error.

Cada editor posee sus propias particularidades, que nada tienen que ver con el lenguaje Basic.

Ejemplo: EDIT 50 (la línea 50, presentada en la pantalla, está dispuesta para ser modificada).

ELSE (Declaración)

Del inglés *ELse* = si no.

Esta declaración va siempre precedida de IF THEN (= si entonces). *Si no se cumple la condición precisada, ejecutar entonces el programa siguiente.*

Ejemplo: IF A = 1 THEN PRINT «UNO» ELSE PRINT «?» (Si la variable A = 1 la pantalla presenta «UNO»; si no, presenta un signo de interrogación).

END (Declaración)

Del inglés *end* = fin.

Esta declaración, que en muchos Basic no es obligatoria, *va al final de un programa.* En algunas versiones puede ir colocada en un lugar cualquiera; tiene entonces el valor de STOP.

ERASE (véase NEW).

F (véase FOR).

FOR (Declaración)

Del inglés *for* = para.

Esta declaración, que forma parte de la declaración completa FOR TO NEXT, *crea un bucle:* las instrucciones comprendidas entre FOR y TO se ejecutan tantas veces como se especifica en TO.

Ejemplo:

```
10 FOR X = 1 TO 10
20 PRINT «BUENOS DÍAS»
30 NEXT X
```

(La expresión «BUENOS DÍAS» se presentará visualmente 10 veces.)

En algunos Basic, es posible *incrementar el paso del bucle* en un valor diferente de 1 con ayuda de la palabra STEP.

Algunos sistemas utilizan la palabra F.

G. (véase GOTO).

GET (véase INYEYS).

GOS (véase GOSUB).

GOSUB ((Declaración)

Del inglés *go* = ir a y *sub* = sub (programa).

Esta instrucción *provoca la transferencia provisional de la ejecución del programa a un subprograma,* el número de cuya primera línea se especifica. Después de la última línea del subprograma, hay que colocar obligatoriamente la instrucción RETURN.

Algunos sistemas utilizan la palabra GOS, en vez de GOSUB.

Ejemplo:

```
10 GOSUB 100
20...
30 ...
100 ...
105 ...
106 ...
110 RETURN
```

(Las líneas 100 a 106 constituyen un subprograma al que se salta gracias a la instrucción GOSUB.)

GOT (véase GOTO).

GOTO (Declaración)

Del inglés *goto* = ir a.

Esta instrucción *obliga al programa a saltar determinadas líneas para posiciones en el número de línea precisado.* Algunos sistemas utilizan la abreviatura G. o la palabra GOT.

Ejemplo: GOTO 100 (cualquiera que sea el lugar en que aparezca esta instrucción, la ejecución del programa queda transferida a la línea 100).

HOME (véase CLS).

I. (véase INT).

IF (Declaración)

Del inglés *if* = si.

Esta declaración se utiliza con THEN *para comprobar una condición.* Si la condición se cumple, el ordenador ejecuta la instrucción situada después de THEN. En caso contrario, pasa a la línea siguiente.

Ejemplo:

```
10 IF A = 1 THEN PRINT «UNO»
20 IF A = 2 THEN PRINT «DOS»
```

(Si la variable A toma el valor 1, en pantalla se visualiza «uno»; si su valor es 2, se visualiza «dos», etc.)

IF se utiliza también con GOTO (IF-GOTO), GOSUB (IF-GOSUB) y LET (IF-LET).

IN. (véase INPUT).

INKEY$ (Función)

Del inglés *in* = en, y *key* = tecla de teclado.

Esta función sirve para *leer un carácter pulsado en el teclado,* tal como lo hace la declaración INPUT. No obstante, a diferencia de esta última, no detiene la ejecución del programa y el ordenador va funcionando hasta que recibe un mensaje procedente del teclado.

Esta función *presenta la ventaja de no modificar la imagen de la pantalla* (cosa interesante, por ejemplo, durante un juego) y de *no necesitar que se pulse la tecla de ejecución* (muy práctico para la ejecución de menús).

Ejemplo: IF INKEYS = «1» GOTO 100 (la pulsación de la tecla 1 provoca un salto a la línea 100).

Algunos ordenadores utilizan la función GET.

INPUT (Declaración)

Del inglés *input* = entrar.

Esta instrucción permite *entrar datos a partir del teclado.* Estos datos se atribuirán a una variable. Cuando el ordenador encuentra INPUT interrumpe el programa, presenta un signo de interrogación y espera datos del teclado.

La variable puede ser numérica o alfanumérica.
Algunos ordenadores utilizan la abreviatura IN.
Ejemplos:
10 INPUT A
20 PRINT A
(La pantalla presenta un signo de interrogación y, a continuación, la cifra pulsada en el teclado.)
10 INPUT A$
20 PRINT A$
(La pantalla presenta un signo de interrogación y, a continuación la cadena de caracteres pulsada en el teclado.)

INT (Función)
Del inglés *INTeger* = número entero.
Esta función se utiliza para *redondear los números a su valor entero* conservando, sin embargo, el signo que precede al número.
Hay que observar que el argumento comprendido entre paréntesis puede ser no solamente *un número,* sino también *una variable* o *una expresión.*
Ejemplo: PRINT INT (1.17) (en la pantalla se visualizará 1).

LET (Declaración)
Del inglés *let* = sea.
Esta instrucción, facultativa en numerosos Basic, *asigna un valor a una variable.*
Ejemplo:
10 LET A = 50 (el valor 50 se asigna a la variable A).

L. (véase LIST).

LI (véase LIST).

LIS (véase LIST).

LIST (Comando)
Del inglés *list* = reproducir una lista.
Presenta en pantalla las líneas de un programa (scrolling). Puede también presentar solamente una o varias líneas haciendo seguir el comando del número de línea deseado o especificando los números del principio y del final.
Algunos dialectos utilizan una de las abreviaturas siguientes: L., LI o LIS.
Ejemplos:
LIST (comando) (presenta visualmente todas las líneas del programa).

LIST 10-50 (presenta las líneas comprendidas entre 10 y 50).
Para detener el listado del programa se pulsa una de las teclas siguientes: STOP, BREAK, CTRL C, etc., en función del sistema empleado.

LLIST (Comando)
Del inglés *Line printer* = impresora, y *list* = reproducir una lista.
Esta instrucción, utilizada como LIST, *hace el listado del programa en la impresora* en vez de que aparezca en la pantalla.

LOAD (Comando)
Del inglés *load* = cargar.
Carga en memoria un programa previamente registrado en un soporte magnético (casete o diskette).
Algunos ordenadores utilizan el comando C(asete) LOAD: CLOAD para distinguir entre diskette y casete.

LPRINT (comando/declaración)
Del inglés *Line printer* = impresora y *print* = imprime.
Este comando es idéntico a PRINT, excepto que la presentación visual en pantalla se sustituye por una impresión sobre papel.

N. (véase NEW o NEXT).

NE (véase NEW).

NEW (véase NEXT).

NEW (Comando)
Del inglés *new* = nuevo.
Borra el programa Basic contenido en memoria y también la pantalla. Este comando no afecta a los programas en lenguaje máquina.
Algunos ordenadores utilizan el comando ERASE o SCRATCH.

NEXT (Declaración)
Del inglés *next* = siguiente. Ver el comando completo FOR TO NEXT.
Algunos sistemas permiten las abreviaturas N. o NEX.

ON GOSUB (Declaración)
Del inglés *on* = sobre, *go* = ir a, y *sub* = sub.

Se comporta como la declaración GOSUB, pero incluye, además, *una comprobación de derivación.*

P. (véase PRINT).

PRI (véase PRINT).

PRINT (Comando/declaración)
Del inglés *print* = imprimir.
Esta instrucción *provoca la visualización en pantalla del valor de las variables* pedidas, las cadenas alfanuméricas, etc.
Puede también emplearse como *un comando directo* y presentar en pantalla el resultado de una operación matemática (como una calculadora).
Algunas veces, esta declaración se completa con la instrucción AT (que ordena imprimir en un determinado lugar) o la instrucción USING (que imprime según un formato preciso).
Algunos ordenadores abrevian esta instrucción con: ?, P. o PRI.

RAN (véase RANDOMIZE).

RANDOM (véase RANDOMIZE).

RANDOMIZE (Declaración)
Del inglés *randomize* = deja al azar.
Se trata de una instrucción doble compuesta de RANDOMIZE y RND.
La declaración RANDOMIZE (que sirve para mezclar números en la memoria) va al principio del programa, mientras que RND (que sirve para reproducir una secuencia distinta a cada llamada) se coloca en función de las necesidades.
Algunos lenguajes utilizan una de las siguientes abreviaturas: RANDOM o RAN.

REA (véase READ).

READ (Declaración)
Del inglés *read* = leer.
Esta instrucción, empleada siempre con la instrucción DATA, lee los datos de una línea DATA y los sitúa en las variables correspondientes.

REM (Declaración)
Del inglés *REMark* = comentario.

Esta declaración sirve para *documentar un programa* (para no perderse en él) colocando discrecionalmente unas observaciones.
Siempre se coloca al principio de línea y el ordenador *no tendrá en cuenta esta línea* en la ejecución del programa.
Algunos sistemas abrevian esta declaración empleando el apóstrofo (').

RESTORE (Declaración)
Del inglés *restore* = restablecer.
Esta instrucción se emplea en un programa que contiene READ y DATA para *reutilizar datos ya leídos* por la función READ.

RETURN (Declaración)
Del inglés *return* = vuelve.
Declaración empleada siempre con GOSUB (o ON GOSUB) para *poner fin a la subrutina.*

RND (Función)
Del inglés *RaNDom* = azar.
Esta función, empleada frecuentemente con RANDOMIZE, crea números de manera aleatoria. La forma en que se emplee variará de un dialecto a otro.

R. (véase RUN).

RU (véase RUN).

RUN (Comando/declaración)
Del inglés *run* = ejecutar.
Provoca la *ejecución del programa en memoria.*
Si se desea que el programa no empiece hasta una línea determinada, *se añade el número de la línea al comando.*
Ejemplo: RUN 5Ø (el programa empezará en la línea 5Ø).
Algunos sistemas emplean la abreviatura RU o R.

SAVE (Comando)
Del inglés *save* = salvar.
Registra en soporte magnético (casete o diskette) un programa contenido en memoria.
Algunos ordenadores utilizan el comando C(asete) SAVE: CSAVE.

SCRATCH (véase NEW).

ST. (véase STOP o STEP).

STE (véase STEP).

STEP (Función)
Del inglés *step* = paso.
Esta función sirve para fijar un incremento distinto de 1 en un bucle FOR NEXT.

STO (véase STOP).

STOP (Declaración)
Del inglés *stop* = paro.
Esta instrucción detiene un programa que se está ejecutando en el lugar escogido.
Puede colocarse en cualquier lugar del programa, excepto en la última línea, en la cual se utiliza END.
Algunos sistemas emplean la abreviatura ST. o STO.

T. (véase THEN).

THE (véase THEN).

THEN (Declaración)
Del inglés *then* = entonces.
Esta declaración se emplea siempre con la declaración IF.
Algunas veces se abrevia así: T o bien THE.

ÍNDICE

OTRAS OBRAS DEL MISMO AUTOR

CÓMO ELEGIR SU MICROORDENADOR

Si es usted de los que todavía no conocen bien los microordenadores, este libro pretende ayudarle a adquirir el que más convenga al uso que quiera darle. Consta de dos partes. La primera está dedicada a la estructura de los principales sistemas informáticos y a todos los accesorios aplicables. La segunda parte es una guía de los microordenadores más difundidos, que le permitirá orientarse en medio de una verdadera jungla de productos.

TRATAMIENTO DE TEXTOS

Un sistema informático para el tratamiento de textos presenta enormes ventajas a todos los que trabajan de manera regular en documentos escritos (secretarias, traductores, periodistas, abogados y otras profesiones liberales).

Este interesante manual le permitirá conocer en qué consiste exactamente el llamado *tratamiento de textos,* para qué sirve, qué es lo que permite hacer, en qué puede serle útil, qué material hay que comprar, y le permitirá también explotar a fondo sus múltiples posibilidades.